■ PROJECTS ■ THEORY ■ APPLICATIONS ■ CIRCUITS ■ TECHNOLOGY

NUTS AND VOLTS

www.nutsvolts.com
June 2015

EVERYTHING FOR ELECTRONICS

THE CRYSTAL PALACE

An Experiment In Serial Bluetooth Control

✦ Fix Up That Old Radio!

✦ Add microSD Capability To Your 32-Bit PIC Project

✦ Build An
- Electronic Lock
- Wireless Freezer Alarm

Make Your Next Design Efficient and Secure

Microchip's PIC24 Family of Devices Includes Low Power, Crypto Engine and Core Independent Peripherals

Core Independent Peripherals found on PIC24 products help offload the CPU to accelerate real-time response and lower power consumption. For example, the hardware crypto engine safeguards embedded data before storage or transfer. With the rapid growth of the Internet of Things, plus the addition of internet connectivity to traditional applications, the need to protect embedded data and extend battery life are essential. The crypto engine found on PIC24 devices ensures data integrity while running completely autonomous from the CPU. The PIC24 family includes industry-leading power consumption with flexible modes and wake-up sources as well as VBAT battery backup capability, precision ADCs, DACs and op amps for easy connection to a wide range of sensors.

- Low-power crypto engine to ensure secure data storage and transfer
- eXtreme Low Power (XLP) technology for longest battery life
- Intelligent Analog integration for easy connection to sensors

www.microchip.com/16bit

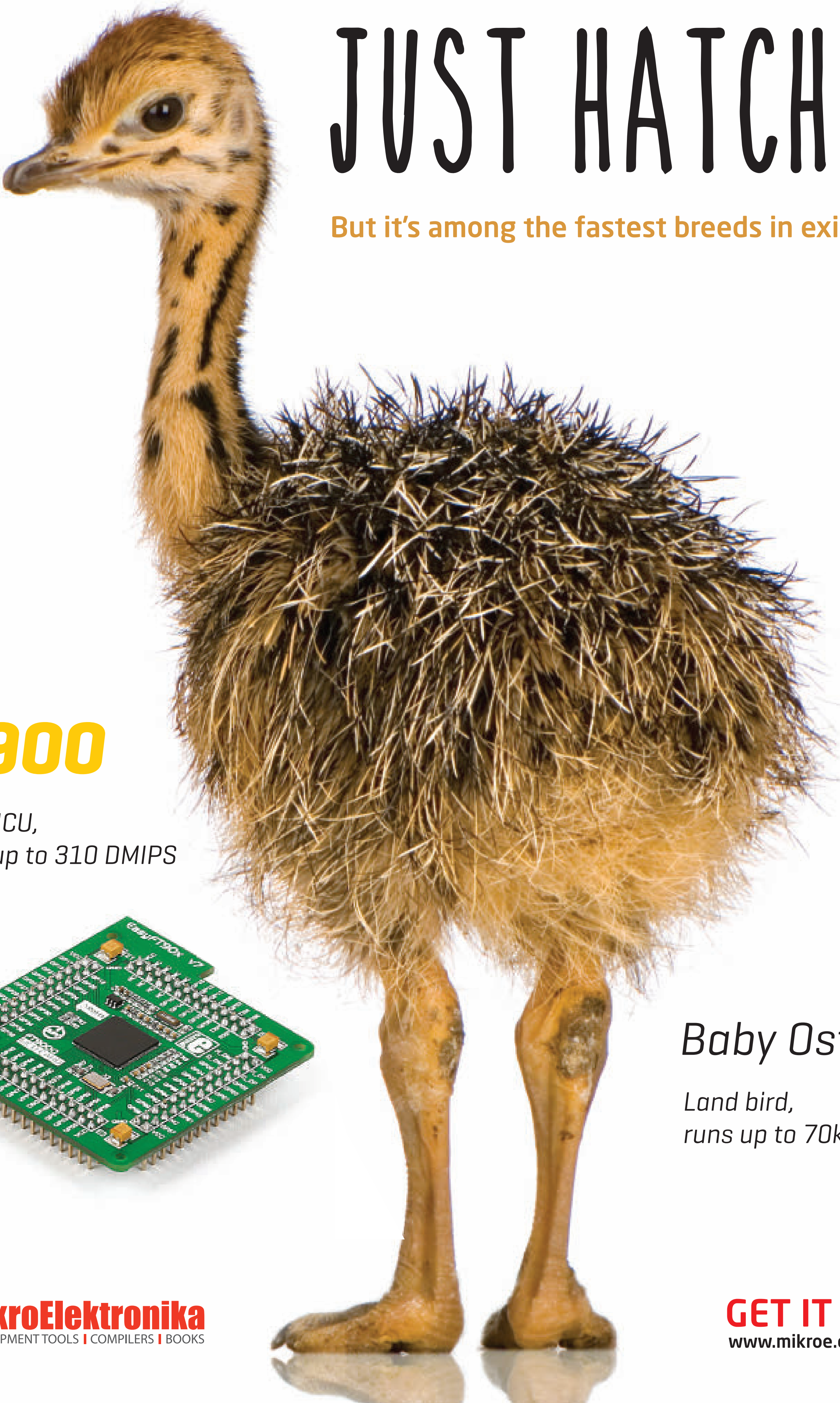
Learn How to Add Low-Power Wireless Connection to Your XLP PIC24 Application

Download AN1861:
Bluetooth® Smart Communication Using Microchip RN4020 Module and 16-bit PIC® Microcontroller

MICROCHIP

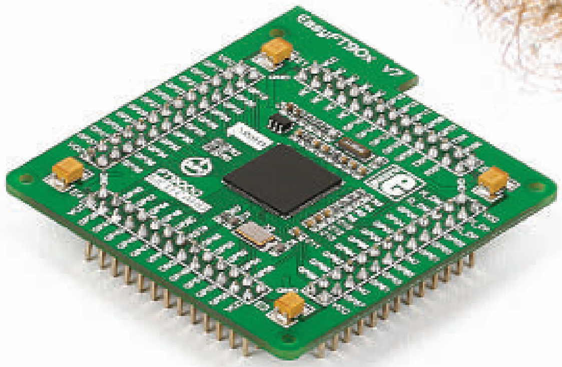
JUST HATCHED

But it's among the fastest breeds in existence



FT900

32-bit MCU,
speeds up to 310 DMIPS



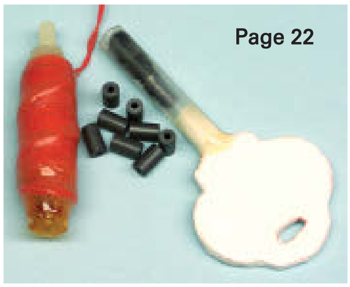
Baby Ostrich

Land bird,
runs up to 70km/h

MikroElektronika
DEVELOPMENT TOOLS | COMPILERS | BOOKS

GET IT NOW
www.mikroe.com/ft90x

June 2015


Page 22

Page 28

Columns

Departments

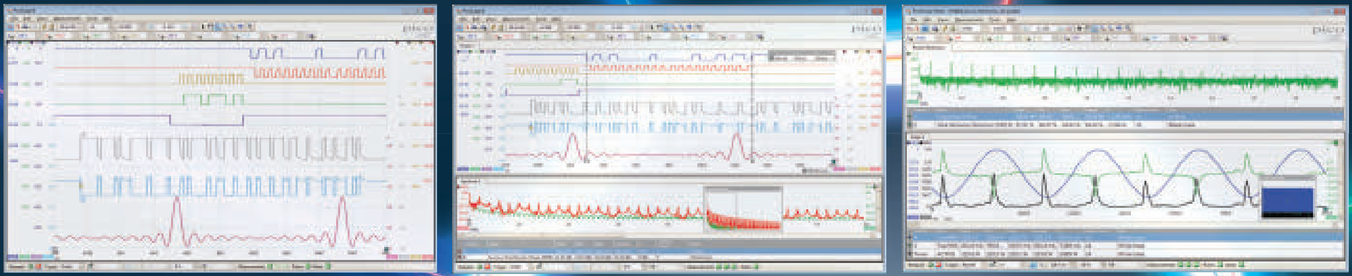
Nuts & Volts (ISSN 1528-9885/CDN Pub Agree #40702530) is published monthly for $26.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL MAILING OFFICES. POSTMASTER: Send address changes to **Nuts & Volts, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@nutsvolts.com.

pico® Technology

7 functions of PicoScope:

1. Oscilloscope 2. Spectrum analyzer 3. Function generator 4. AWG
5. Logic analyzer 6. Serial protocol analyzer 7. Automatic waveform test

8 CHANNEL PC OSCILLOSCOPE



For just $2300

- High resolution
- USB powered
- Deep memory

INCLUDES AUTOMATIC MEASUREMENTS, SPECTRUM ANALYZER, SDK, ADVANCED TRIGGERS, COLOR PERSISTENCE, SERIAL DECODING (CAN, LIN, RS232, I²C, I²S, FLEXRAY, SPI), MASKS, MATH CHANNELS, ALL AS STANDARD, WITH FREE UPDATES

12 bit • 20 MHz • 80 MS/s • 256 MS buffer • 14 bit AWG

Call Us At: 1-800-591-2796   www.picotech.com/pco541

DEVELOPING PERSPECTIVES

Going Beyond Shallow Understanding

Thanks to readily available kits, DIY articles, and web resources, it's a simple matter to cobble together a functional circuit with little real understanding of the underlying electronics. The circuit description for an audible siren kit might read something like "Q1 and Q2 form an astable multivibrator." At some level, this may be adequate. However, if you're interested in truly understanding an astable multivibrator — or any other circuit for that matter — you have to dig deeper.

There's a cost, of course, for digging deeper. You have to invest the time to read about various oscillator circuits including the tradeoffs of each design, perhaps historical uses of the circuit, and perhaps applications beyond that of a siren. The payoff for going beyond the surface descriptions of circuits on a regular basis is the ability to intuit circuits. It's a skill that I'd say is possessed by less than 1% of hobbyists.

Developing a deeper understanding of electronics often means looking into other systems. For instance, if you gain an understanding of a mechanical oscillator, many of the principles will transfer to an electronic version. Consider that mechanical damping has direct parallels with electric dampers, for example. In this regard, digging deeper often involves studying the history of engineering.

My favorite historical topics are watches, clocks, and robotics — all of which predate electronics by over a century. By searching the Web — including the US Patent and Trademark Office site — I've found mechanical analogs for everything from batteries (springs) and voltage regulators (mechanical regulators) to ergonomic user interfaces (watch faces and winding stems). In some cases, I find myself wondering why perfectly good mechanical designs were replaced by inferior electronic circuits. I also wonder if there is anyone alive today capable of recreating the mechanical systems of a century ago. Of course, if you're not comfortable with self-directed learning, there are schools of

JOIN TEAM SYNERGY MOON!

Synergy Space Explorers are the power that drives the Synergy Moon space program. We are Team Synergy Moon, building a mission to the moon that includes Micro Satellites, Lunar Rovers and a Lunar Lander. We're going into space this year, and we're going to the moon with our Google Lunar XPRIZE mission.

GOOGLE LUNAR XPRIZE SYNERGY MOON OFFICIAL TEAM

You will have the opportunity to participate in space research, exploration and development missions, which currently include our Google Lunar XPRIZE mission to the moon, the Artemis NanoSat Constellation project and our remote controlled Tesla Orbital Space Telescope.

Get on board now and be part of this great adventure!

DIY SATELLITES AND SPACECRAFT SYSTEMS
info@synergymoon.com

NUTS AND VOLTS
EVERYTHING FOR ELECTRONICS

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct.
Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore orders only **1-800-783-4624**
www.nutsvolts.com

Subscription Orders
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277
North Hollywood, CA 91615

FOUNDER
Jack Lemieux

PUBLISHER
Larry Lemieux
publisher@nutsvolts.com

ASSOCIATE PUBLISHER/ ADVERTISING SALES
Robin Lemieux
robin@nutsvolts.com

EDITOR
Bryan Bergeron
techedit-nutsvolts@yahoo.com

VP OF OPERATIONS
Vern Graner
vern@nutsvolts.com

CONTRIBUTING EDITORS
Fred Eady, Tim Brown, Jon McPhalen, Jeff Eckert, Paul Verhage, Chuck Hellebuyck, Andrew Retallack, J.W. Koebel, Eric Bogatin, Craig Lindley, Ron Newton, Devlin Gualtieri

CIRCULATION DEPARTMENT
subscribe@nutsvolts.com

SHOW COORDINATOR
Audrey Lemieux

WEB CONTENT
Michael Kaudze
website@nutsvolts.com

WEBSTORE MARKETING
Brian Kirkpatrick
sales@nutsvolts.com

WEBSTORE MANAGER
Sean Lemieux

ADMINISTRATIVE STAFF
Debbie Stauffacher
Re Gandara

engineering that can give you a formal education in electronics. Whether or not that translates to the ability to intuit electronic circuits depends on the school and how you leverage your experience. That said, I've found hands-on experience the best teacher regardless of the learning environment. You have to get your hands dirty to truly understand electronics. So, get that soldering iron ready. **NV**

Feedback Motion Control

The Old Way
1) Build robot
2) Guess PID coefficients
3) Test
3a) Express disappointment
3b) Search Internet, modify PID values
3c) Read book, modify PID coefficients again
3d) Decide performance is good enough
3e) Realize it isn't
3f) See if anyone just sells a giant servo
3g) Express disappointment
3h) Re-guess PID coefficients
3i) Switch processor
3j) Dust off old Differential Equations book
3k) Remember why the book was so dusty
3l) Calculate new, wildly different PID coefficients
3m) Invent new, wildly different swear words
3n) Research fuzzy logic
3o) Now it is certainly not working in uncertain ways
3p) Pull hair
3q) Switch controller
3r) Re-guess PID coefficients
3s) Switch programming language
3t) Start a new project that doesn't need feedback control
3u) See parts in box. Feel guilty. Go back to old project
3v) Start testing every possible combination of PID coefficients
3w) Apply eye drops to red, bleary, sleep-deprived eyes
3x) Wait, it's working!
3y) Decide not to do any more projects that require control systems
3z) Wonder why someone doesn't just make a thing that tunes itself

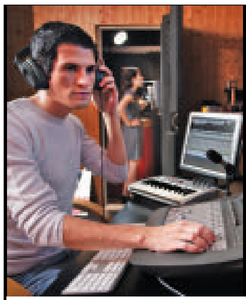
The Kangaroo x2 Way
1) Build robot
2) Press Autotune
3) Get a snack

Kangaroo x2 adds self-tuning feedback to SyRen and Sabertooth motor drivers.

$24.99

DimensionEngineering
www.dimensionengineering.com/kangaroo

Make up to $100 an Hour or More!



Be an FCC LICENSED ELECTRONIC TECHNICIAN!

An "FCC Commercial License" is your ticket to thousands of high paying jobs:
- Radio-TV • Communications • Avionics
- Maritime • Radar • Municipal and more!

The United States of America
Federal Communications Commission
GENERAL RADIOTELEPHONE OPERATOR LICENSE

This proven Home-Study Course prepares you to get your FCC License at your own pace, in your own home -- *GUARANTEED!*
- No costly school
- No classes to attend
- Easy, fast, low cost
- No experience needed

You get your FCC License or your money back! Thousands of successful graduates!

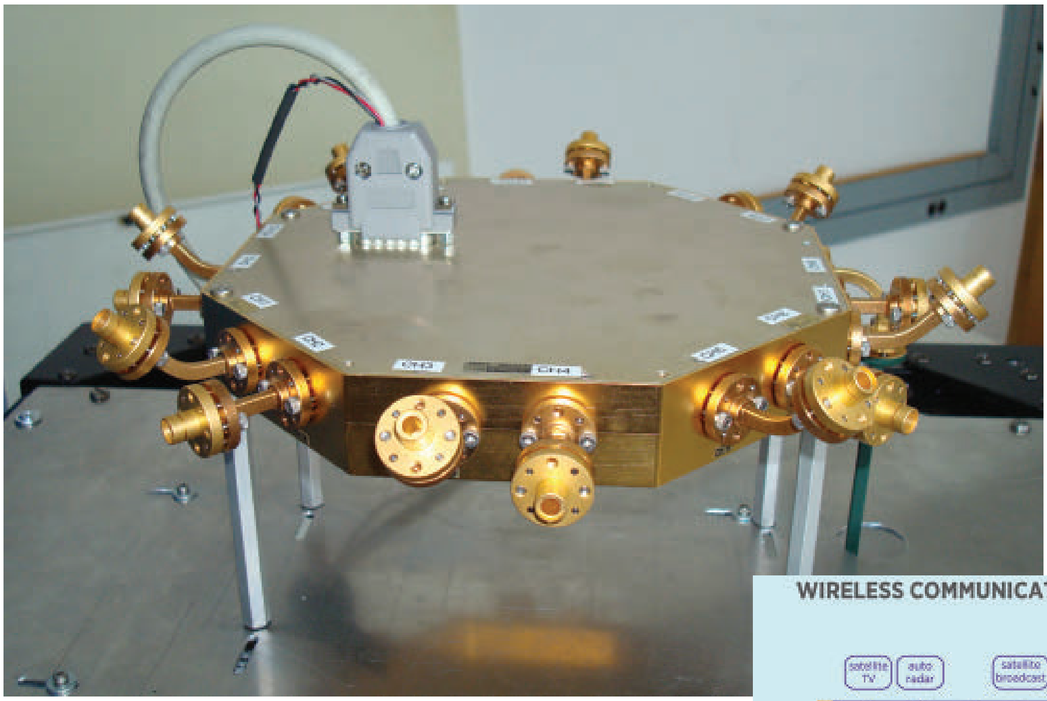
***Call for FREE info kit:* 800-877-8433** ext. 109

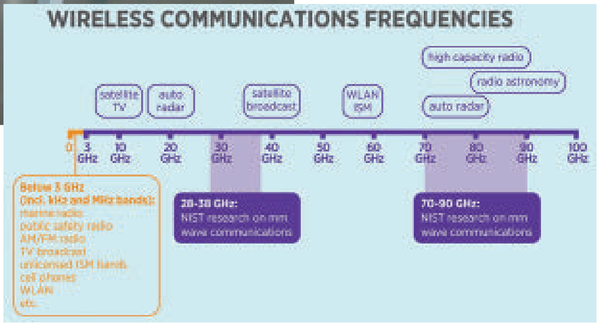
or, visit: **www.LicenseTraining.com**

COMMAND PRODUCTIONS • FCC License Training
Industrial Center, 480 Gate Five Rd., PO Box 3000, Sausalito, CA 94966-3000

Our 39th Year

ADVANCED TECHNOLOGY


■ Directional 16-antenna array used for modeling of wireless communications channels at 83 GHz.


WIRELESS COMMUNICATIONS FREQUENCIES
high capacity radio
radio astronomy
satellite TV
auto radar
satellite broadcast
WLAN ISM
auto radar
0
3 GHz
10 GHz
20 GHz
30 GHz
40 GHz
50 GHz
60 GHz
70 GHz
80 GHz
90 GHz
100 GHz
Below 3 GHz (incl. kHz and MHz bands): marine radio, public safety radio, AM/FM radio, TV broadcast, unlicensed ISM bands, cell phones, WLAN, etc.
28-38 GHz: NIST research on mm wave communications
70-90 GHz: NIST research on mm wave communications

Gearing Up for Millimeter-Wave Communications

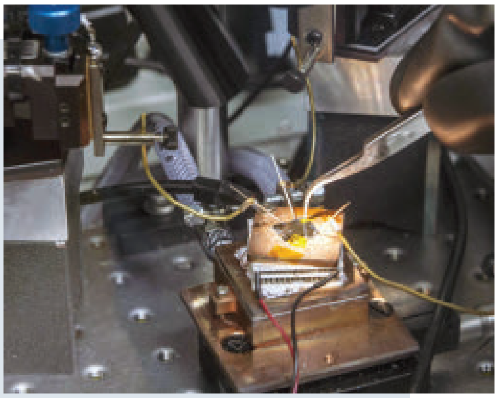
It's fairly well known that we're running low on space in existing wireless communications frequencies, and the DC through microwave spectrum is pretty much used up. Researchers worldwide are looking at gigahertz signals to create more channel capacity, as higher frequencies permit higher data rates. According to the laws of physics, however, at any particular power level, shorter wavelengths translate into shorter range. Plus, transmissions in the millimeter band are subject to higher atmospheric attenuation, rain fade, and other problems. According to the National Institute of Standards and Technology (NIST, **www.nist.gov**), "Because high speed digital circuits can easily distort millimeter-wave signals, even tiny errors can result in erroneous bits of information. In addition, millimeter waves don't travel around corners as well as lower frequency waves, so channel models will be complex. Possible solutions include development of complex antenna arrays that may provide novel capabilities such as beam steering — the capability to transmit in many different directions to point the beam directly at the receiving device, and even track mobile devices. This would strengthen signals and cause less interference to neighboring devices."

To address such issues, NIST folks are developing measurement tools for new mobile channels that could offer more than 1,000 times the bandwidth of today's cell phones. So far, they have developed a calibrated modulated signal source to test millimeter-wave instruments such as receivers, and channel sounders for both indoor and outdoor environments. The source has been demonstrated at 44 and 94 GHz, giving signal measurements traced to fundamental physical quantities. The mobile channel sounder — demonstrated at 83 GHz — provides calibrated received signal strength and other data for analyzing signal scattering and reflections. NIST has also developed an advanced probe designed for fields above 100 GHz.

"We want to provide US industry with the precision measurement methods needed to develop innovative millimeter-wave wireless technologies and associated standards," NIST project coordinator Kate Remley noted. "This work can advance the state-of-the-art in telecommunications and help meet the expected increases in demand for wireless capacity." ▲

ADVANCED TECHNOLOGY Continued

Not Just for Metals Anymore


■ Semiconductor devices being electrically bonded with new solder.

If most of us made a list of technologies that pique our interest, solder would be pretty near the bottom. Sure, there are many alloys with many different properties, but the basic function is the same: to join metal to metal. Until recently, joining pieces of semiconductor material has been considered impossible, as semiconducting surfaces are so sensitive to impurities and structural defects that the junctions will block the flow of electricity. Current semi manufacturing, therefore, is designed to require no soldering: You just grow a crystal, cut it up, and etch the desired shapes. Now, a team at the University of Chicago (**www.uchicago.edu**) has come up with a compound of cadmium, lead, and bismuth that can be applied as a liquid or paste to join chunks of semiconductor just by heating them to several hundred degrees Celsius.

According to Prof. Dmitri Talapin, "Our paste or our liquid converts cleanly into a material that will be compositionally matched to the bonded parts, and that required development of new chemistry. We had to design special molecules that fulfill this requirement so that they do not contaminate the material."

After application as a liquid or paste, they decompose to form a seamless joint. Talapin cautioned that semiconductor soldering may have little impact on today's technology, but it could lead to the development of cheaper solution-processed semiconductors for new markets, including 3D printing, printable electronics, flat-panel display manufacturing, solar cells, and others. It was noted that this technique has set a new record for electron mobility in solution-processed semiconductors (an indication of how quickly electricity flows through materials). This new record is nearly ten times faster than the previous one, which Talapin called "mind blowing." ▲

COMPUTERS and NETWORKING

Mainframe Handles 2.5 Billion Transactions Daily

■ IBM's latest mainframe, the z13.

The market for mainframe computers has been shrinking for a long time, which may be why IBM (**www.ibm.com**) has been bringing out new models at a fairly leisurely rate. Big enterprises with big processing needs (airlines, banks, and so on) still rely on them, so the company still happily provides. The latest is the z13, designed for the emerging mobile economy. Among its cited attributes are the ability to process 2.5 billion transactions per day, real time encryption of all mobile transactions, and embedded analytics that provide real time "transaction insights" 17 times faster than competitive systems.

The latter provides a big boost to transaction security. According to IBM, "The z13 features the world's fastest microprocessor — two times faster than the most common server processors, 300 percent more memory, 100 percent more bandwidth, and vector processing analytics to speed mobile transactions. As a result, the z13 transaction engine is capable of analyzing transactions in real time and will be able to help prevent fraud as it is occurring, allowing financial institutions to halt the transaction before the consumer is impacted. IBM has designed the z13 to integrate real time scoring and guarantees this capability as a feature of the system. This scoring can be used for fraud detection on 100 percent of a client's business transactions."

IBM isn't big on providing detailed specs and, in fact, notes that clock rate is no longer the primary means of achieving high performance. Nevertheless, we did determine that the maximum frequency is 5 GHz — a bit slower than the previous z12 model. However, the z13 is available with as many as 141 configurable processor units, supporting up to 8,000 virtual servers in one footprint. The machine has been installed in a few locations as of this writing, with more slated for coming months. ▲

Post comments on this article at **www.nutsvolts.com/index.php?/magazine/article/june2015_TechKnow15**.

COMPUTERS and NETWORKING Continued

IoT Starter Kit in the Offing

At the lower end of the computing spectrum is the venerable ARM processor, originally introduced in the 1980s. (Trivia item: ARM stands for Acorn RISC Machine, originally developed by England's Acorn Computers.) One of the latest offerings from ARM Ltd. (**www.arm.com**) is the ARM® mbed™IoT Starter Kit-Ethernet Edition, designed to channel data from Internet-connected devices directly into IBM's Bluemix cloud platform. The kit consists of a development board from Freescale Semiconductor (**www.freescale.com**), an ARM Cortex®-M4 based processor, and a sensor I/O application shield. According to ARM, future versions will run the new ARM mbed OS and utilize ARM mbed device server software to deliver a wider range of security, communication, and device management features. ARM and IBM will continue to work together on interoperable, open, secure, and scalable connectivity between devices and the cloud.

On the IBM side of the venture, Meg Divitto, VP for IoT, noted, "The Internet of Things is about bringing the physical and digital worlds closer together, to allow businesses to better understand and interact with what is happening around them. In order to make this work for businesses, it needs to be simple to connect physical devices into the cloud and to build applications and insights around them. IBM Bluemix and the new ARM mbed starter kit are designed to substantially enhance that effort."

As of this writing, the prototypes of the kit have been provided to some early adopters, but it is not yet available to the rest of us. No price information has been released, either. By the time you read this, though, it should be on the shelf at the usual suppliers. ▲

■ The soon-to-be-available ARM® mbed™ IoT Starter Kit.

CIRCUITS and DEVICES

Wise Up Your Vehicle

It's pretty tough to find positive reviews of GM's OnStar, which may be why Verizon Vehicle (**www.verizonvehicle.com**) is starting up a competing service this spring. Come to think of it, it's pretty tough to find positive reviews of Verizon, but VV is the direct result of the 2012 acquisition of Atlanta-based Hughes Telematics, so that may not matter. Verizon paid pocket change — $612 million — for the company, but it was actually losing money and worth only about half that amount on the books. However, maybe things will work out.

■ The Verizon Vehicle Bluetooth speaker unit.

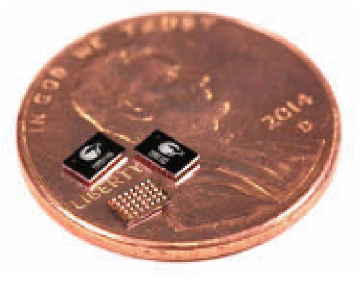
The good news is that VV could allow you to connect your clunky old Internet-incapable rust bucket to communicate with the outside world — at least if it was built after 1996 and has an onboard diagnostic (OBD) port. Features include GPS-directed roadside assistance, an automatic "urgent incident" alert system, one-button SOS assistance for emergencies, and an auto health system with predictive diagnostics.

You also get alerts when scheduled maintenance is required, stolen vehicle location assistance, and even parking tools (via the smartphone app) that help you find your way back to the car and keep track of how much time is left on the parking meter. We're looking at $15 per month for the service, which is considerably cheaper than OnStar. ▲

CIRCUITS and DEVICES Continued

Controller Enables USB 3.1 Type-C

Okay, it is highly unlikely that you own anything with a USB Type-C port, but someday you will. One of its best features is that it's reversible, so you will no longer plug a USB cable in upside-down 50 percent of the time. Plus, USB 3.1 provides data transfer speeds up to a theoretical 10 Gbps and the ability to draw up to 100W of power. Helping to enable the technology is Cypress Semiconductor's (**www.cypress.com**) new CCG1 port controller family, presently available in sampling quantities. Based on the PSoC® 4 programmable system-on-chip architecture, it integrates a low power ARM® Cortex™M0 core (the smallest available ARM processor) with PSoC's proprietary programmable analog and digital peripherals. CCG1 is available in a 40-pin QFN for notebook applications, a 16-pin SOIC and a 28-pin SSOP for power adapters, and a 35-ball WLCSP for cable and mobile applications. ▲



■ Cypress' CCG1 USB programmable USB controller.

Look Young Again!

Last — and certainly least — that inquiring minds need to know about is the Facial Lift At Once Face Trainer Alpha, described as a "simple but effective way to fight the signs of aging around your face." The Alpha version is actually an update of the "previous bestseller," but is now waterproof so you can use the device in the bathtub (although you are cautioned not to submerge it).

All you have to do is select among several levels of 360° pulsation, shove it into your mouth, and keep it there for three minutes every day while making sure you avoid contact with your teeth. This allows the device to "push and work on your facial muscles little by little every time, 'training' your facial expression so you will look and feel younger." To get this electrical fountain of youth, just log onto **www.japantrendshop.com** and pay $92 (plus $14 shipping). ▲

■ The Facial Lift At Once Face Trainer Alpha.

INDUSTRY and the PROFESSION

Best Bang for Your Solar Buck

When considering the installation of a commercial or residential solar energy system, one of the fundamental decisions is whether to finance the equipment or lease it from a third-party owner. Fortunately, the US Department of Energy's National Renewable Energy Laboratory (NREL) has done the research for us and offers two free reports on the subject. The first one, "Banking on Solar: An Analysis of Banking Opportunities in the US Distributed Photovoltaic Market," provides a high-level overview of the developing US solar loan product landscape. The analysis covers the range of consumer and commercial loan products available for financing solar in the United States, discusses the potential and active market players in the distributed solar loan space, and provides qualitative and quantitative analyses of how solar loans of varying maturities stack up against third-party financing.

The second report, "To Own or Lease Solar: Understanding Commercial Retailers Decisions to Use Alternative Financing Models," examines the tradeoffs between financing methods for businesses installing on-site PV systems. To download PDF versions of the reports, visit **www.nrel.gov/publications**. **NV**

Q & A

WITH TIM BROWN

In this column, Tim answers questions about all aspects of electronics, including computer hardware, software, circuits, electronic theory, troubleshooting, and anything else of interest to the hobbyist. Feel free to participate with your questions, comments, or suggestions. **Send all questions and comments to: Q&A@nutsvolts.com.**

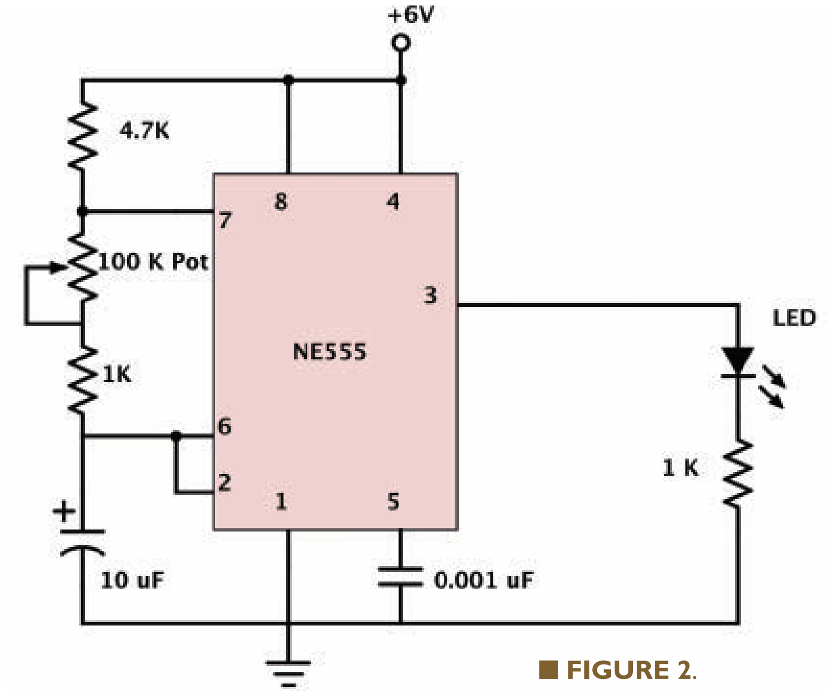
Help Finding Components

Q In reference to the article, "A Light House for Short People" from the 1968 Winter Edition of *Electronics Experimenter's Handbook* by *Popular Electronics*, I would like to build the lighthouse for my granddaughter. Can you suggest some easier to find components? I do have some neon bulbs on hand.

— Dr. Jeff Helgoe
Ladysmith, WI

A I always enjoy looking back at where we have been in electronics, and Jeff's article from the 1968 *Experimenter's Handbook* is certainly a stroll down memory lane for me. *Popular Electronics* started their publication in October 1954 and continued under different publishers until December 1999. I read some of my uncle's magazines from the late '50s and on my migration to *Nuts & Volts*, I subscribed to *Radio Electronics* which switched me over to *Popular Electronics* for a little while in the late '90s before I was traded to *Nuts & Volts* (in the end I think I got the best deal in the trade).

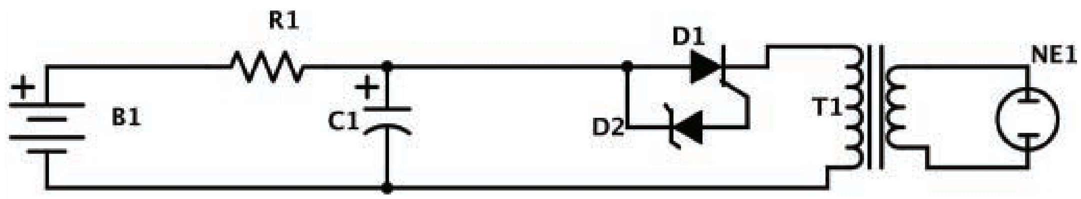
I redrew the 1968 lighthouse circuit (see **Figure 1**) which used a nine volt transistor battery, GE Type C6U SCR, 1N754 zener diode, NE-2 neon lamp, 100 µF/12V capacitor, 2.6 KΩ resistor, and Argonne AR-110 transformer wired to step up the voltage. I was able to locate suitable parts for everything except the AR-110. From my past experience, it is difficult at best to find information on transformers. I suspect that the AR-110 with its 10 KΩ primary and 16Ω secondary (25-to-1 turns ratio) was used as an impedance matching transformer.



■ FIGURE 2.

Argonne transformers still exist, but this particular model apparently is obsolete (maybe one of our readers can help here). Also, I could not find any transformers with turn ratios as high as 25-to-1 (plenty of 10-to-1s). I suspect the advent of ICs has something to do with this.

As they say, "necessity is the mother of invention" (in my case, maybe desperation), so I came up with an LED flasher using an NE555 IC timer which is shown in **Figure 2**. If you use a super bright LED, you may need to adjust the LED's current-limiting resistor to achieve the correct current to give the best illumination. I show a six volt power supply, but the NE555 can operate from 4.5V to 16V, so the nine volt battery may be a better choice than four alkaline batteries.



■ FIGURE 1.

Post comments on this article at **www.nutsvolts.com/index.php?/magazine/article/june2015_QA.**

Ice Melt Tape Controls

Q Do you have any diagrams from past articles that I could use to control roof ice melt tape? I would need to control it by temperature, and perhaps have a timer set to turn off after a few hours, but still turn on if need be. Also, how would I obtain this article?

— Ronald Clerici
via email

A I live in the sunny South, so I'm not exactly woefully qualified to answer questions involving snow. The "storm of the century" back in 1993 dropped only four inches (maybe) on our area. Our biggest problem is the ice that hits in February and March that takes out power lines. I am thankful for gas logs and I have used the pipe heating tape to prevent freezing of water pipes. I've lived and traveled in "snow zones" during winter and I own some arctic clothing, so I do know a little about your problems.

My understanding of the use of roof de-icing cable is to eliminate ice dams that form at the edge of the roof and gutter system when water from snow was melted by the sun in the daytime, then refroze at night. An effective roof de-icing system should sense both outdoor temperature and the presence of moisture (with no moisture present on a cold night, there is no sense in wasting heat/electricity). A de-icing control also should be able to withstand the local environment (heat, cold, moisture, animals) and be safety rated. A "homebrew" de-icing control would use a pair of wires feeding an op-amp to detect moisture and a temperature sensing circuit (use a microcontroller) encased in a weatherproof enclosure.

Looking at cost, safety, and effectiveness of operation, I would recommend a commercially available (UL rated) unit matched to the heat cable. One such unit is the EasyHeat RS-2 Roof Cable De-icing Control which senses outside temperature, the presence of moisture, and can control up to 240 feet (1,200 watts) of EasyHeat cable. The cable should be installed along the edges of the roof, in valleys, gutters, and downspouts, so the number of controllers and length of cable needed depends on the dimensions of your house and gutter system. Check out the EasyHeat website at **www.emersonindustrial.com/en-us/egselectricalgroup/products/heating-cables/residential-heating-cables/residential-roof-gutter-de-icing/Pages/default.aspx**.

I know there are many readers who were hoping for a "roll your own," solution but when it comes to the potential damage due to failing to properly remove the snow/runoff and the possibility of burning down your house (not to mention running afoul of the insurance companies and local jurisdictions), I prefer a solution that has been tested and proven before I buy and install it. REMEMBER, these roof cable de-icing systems (as do all electrical devices operating around water) need to be plugged into a GFCI to protect people from electrocution.

I hope this information helps you to protect your home from the effects of snow and icing. **NV**

MAILBAG

Re: Steven's (KJ6STF) reply in the February 2015 Mailbag, on measuring current with a clamp-on ammeter:

I would add — as a professional commercial electrician and industrial computer service co-owner — I used to be confronted by many problematical power supply/noise surge situations. Out of need, I created a very compact and useful tool by using an extension cord which had one or the other damaged ends. By cutting off the bad end and rolling the cord under foot or on a flat board with a foot to loosen up the insulated wires within, I would slit the outer jacket near the remaining good end and slip out the black conductor long enough to make a 10-turn/2" or so dia coil.

Wire tie the coil and remaining cord body together, and then pull back any remaining slack by cutting the cord to a suitable length, then install the appropriate cord end as needed. Separate five of the turns and carefully wrap/bind them with tape, leaving the remaining coil turns unbound.

By plugging the load into and through this cord, one can use 1, 1+1, 1+2, 1+3, 5, or 5+1 turns, etc., as current multipliers with an Amprobe to get very accurate low current values (from 1 to 11 times the actual current value on the lowest current range) without any problem or additional equipment needed (or at any out of pocket expense). I always have this tool with me, and it has saved the proverbial "pig meat" many times for me! Hope it's useful.

James Kirkendall
via email

Thanks James, for this great advice. I will add it to my tool bag. You have created a variable turns transformer which, in essence, multiplies the current delivered to the Amprobe. It is vitally important to remember that you use the same conductor where the current is flowing in the same direction at the same time in this improvised current transformer because if you happen to mix conductors, their currents will cancel each other's signals. This is another example of the power of our readers to develop unique solutions to perplexing problems. Keep up the good work!.

Tim Brown

■ BY JON MCPHALEN

Tag 'Em, Danno!

The spring is a very busy time for me, filled with travel and lots of extra activities. While on my travels, I had conversations with two different people — just a week apart — having to do with using an RFID reader with the Propeller. The great thing about RFID is that the tags are unique keys. Hence, must be present to operate. Passive RFID tags are very popular devices for access control, and that's what we're going to talk about this month.

Timing is Everything

Before we jump into using the Parallax RFID reader, let's spend a few moments on timing. The funny thing is that while we all ask for more speed, oftentimes we end up inserting delays into programs to throttle them back. There's nothing wrong with this, of course, but sitting in a **waitcnt** means that we're not doing work during that time.

Truth be told, I'm pointing the finger squarely at myself. With the resources available in the Propeller, we often don't think about doing more than one thing in a cog. Again, this is okay, but of late I've been writing programs that are somewhat involved, and I'm now being a bit stingy vis-à-vis cog use. One of my current projects uses all eight cogs, and I still need to time a bunch of discrete events.

This led me to adopt a strategy that's been around forever, and to create an object to simplify timing in my projects. What I'm tending to do now is let the main loop run without any delays (aside from that caused by calling other methods). For an event that I'd like to run at a specific interval, I create a little timer and let it determine when to do something.

Let's look at a simple blinker — the old way:

```
pub main | t

  t := cnt
  repeat
    io.toggle(LED_26)
    waitcnt(t += (100 * MS_001))
```

This code will toggle the LED on P26 of the Activity board every 100 ms. Even if we want to change the duty cycle away from 50%, it's no problem:

```
pub main | t

  t := cnt
  repeat
    io.high(LED_26)
    waitcnt(t += (100 * MS_001))
    io.low(LED_26)
    waitcnt(t += (400 * MS_001))
```

See? Easy peasy. Then, one of our cheeky friends asks us to blink two discrete LEDs at wildly different rates. "Aha!," we shoot back, "We can just put the blinker code in a cog and call it twice!":

```
pri blinker(pin, on_ms, off_ms) | t

  t := cnt
  repeat
    io.high(pin)
    waitcnt(t += (on_ms * MS_001))
    io.low(pin)
    waitcnt(t += (off_ms * MS_001))
```

"Boom, baby! Take that!" After a while, it finally hits us: Do we really want to chew up an entire cog to blink an LED? Of course we can, but there will come a time when we just don't have the resources to do that.

Luckily, the Propeller has a free-running counter (the ***cnt*** register) that we can use for timing (**waitcnt** looks for a specific value in ***cnt***). The trick with **waitcnt** is that it runs in system ticks (hence, not great for long periods) and is signed.

Cutting to the chase, I created a very simple — yet very useful — little object that takes care of the time difference between access calls to the object's methods. Let's have a look at it in action:

```
pub main

  io.low(LED_26)
  led26tmr.start

  io.low(LED_27)
  led27tmr.start

  repeat
    update_p26
    update_p27


pub update_p26

  if (led26tmr.millis => 125)
    led26tmr.adjust(-125)
    io.toggle(LED_26)
```

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/june2015_SpinZone**.

BOM

ITEM	SOURCE/PART #
Propeller Activity Board	Parallax #32910
Serial RFID Reader	Parallax #28140
Servo Extender Cable	Parallax #800-00120
	EFX-TEK #805-00035 (short)
	EFX-TEK #805-00175 (long)
RFID Tags	**Parallax.com**

```
pub update_p27

  if (outa[LED_27] == 1)
    if (led27tmr.millis => 333)
      led27tmr.adjust(-333)
      io.low(LED_27)
  else
    if (led27tmr.millis => 667)
      led27tmr.adjust(-667)
      io.high(LED_27)
```

In this case, we're going to set up each of the LED pins (making them outputs and low) and start a timer for each. Notice the simplicity of the loop in ***main***: All it has to do is call the methods that process each of the LEDs.

For the LED on P26, we want it to toggle every 125 milliseconds. By calling the ***millis()*** method, we can tell when it's time to toggle. Note the use of => when checking the timer versus ==. Why? Well, the world is an imperfect place. If some other process takes a bit more time that expected, we might not hit dead on the delay value using ==; we've all seen this when missing a **waitcnt** target. Using => ensures the toggle at or just after the target delay.

When the switch time arrives, we back up the timer with the ***adjust()*** method; this simplifies checking by not having a moving target in the ***millis()*** value, and it also allows the next change to take place when it should. The LED gets updated to the new state and we're out. Note that each of the LED process methods are very short, hence, won't take much time — this leaves time for the main loop to do other things.

In the past, I've used a specialty LED blinker cog to provide a simple visual indication of a program's state. For example: The LED is off when there's no problem; a short blink for a small problem; and a long blink for a big problem. Here's how we can tackle this requirement without using a cog:

```
pub update_status_led

  case state
    0:
      io.low(LED_26)
      ledtimer.start

    1:
      if (outa[LED_26] == 1)
        if (ledtimer.millis => 100)
          ledtimer.start
          io.low(LED_26)
      else
        if (ledtimer.millis => 900)
          ledtimer.start
          io.high(LED_26)

    2:
      if (ledtimer.millis => 500)
        ledtimer.start
        io.toggle(LED_26)
```

In this case — where the timing changes from state to state — using ***adjust()*** could be problematic, so the timer is simply restarted.

What I'm doing here is not new or unique; programmers have been coding like this for decades. Still, in our *"Keep It Simple, Sam!"* world view, we often resort to strategies that — while simple — are inefficient and can limit what a program does. To the degree it's possible in my projects, I'm banishing **waitcnt** and ***pause()***!

Is That You?

As I stated at the top, I've had recent interactions with people that want to connect the Parallax serial RFID reader to a Propeller. "*No problem!*" I told them — then I thought I should review my own code.

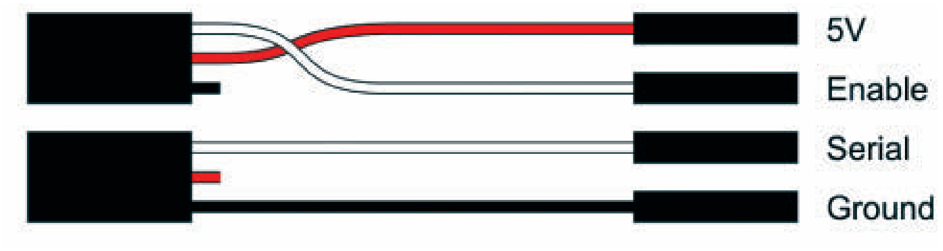
In the past, I've written very simple code that only demonstrates how to get data from the reader, display it, and compare it to a list of known tags — the kind of thing one might do for an access control system. The code works fine, though I've improved on searching the list of known tags. Still, I wanted to create a framework that was a little more flexible, and my nifty time object helped out.

Here's where I'm going: When the RFID reader is enabled, it draws a fair bit of current. Since the presentation of tags is sporadic at best, there's no need to leave the reader on all the time. The other side of that is we don't want to have to do anything special to present the tag — other than holding it up to the reader.

To me, all of this is a very low speed human activity. So, I decided that the reader — when waiting on a tag — would be activated for one second, then deactivated for another second. This will save energy and a one second delay to a person with an RFID tag will cause no problems. In fact, I think we're all used to holding an RFID tag to the reader for a couple seconds before a response is expected.

Honestly, my first thought was to create a PASM object to do the work of enabling the reader and waiting for serial data. After a few cups of coffee and a walk around the block to clear my head, I changed my mind. We already have good serial code, so why re-invent that wheel — especially when so many projects will use the same serial object, anyway.

Note to new Propeller users: When the same code (file) is used for multiple objects, only one copy of the object code is loaded into hub RAM — this maximizes available space. If two or more objects are declared using the same code, each object will get its own set of variables, but code and data will be shared.

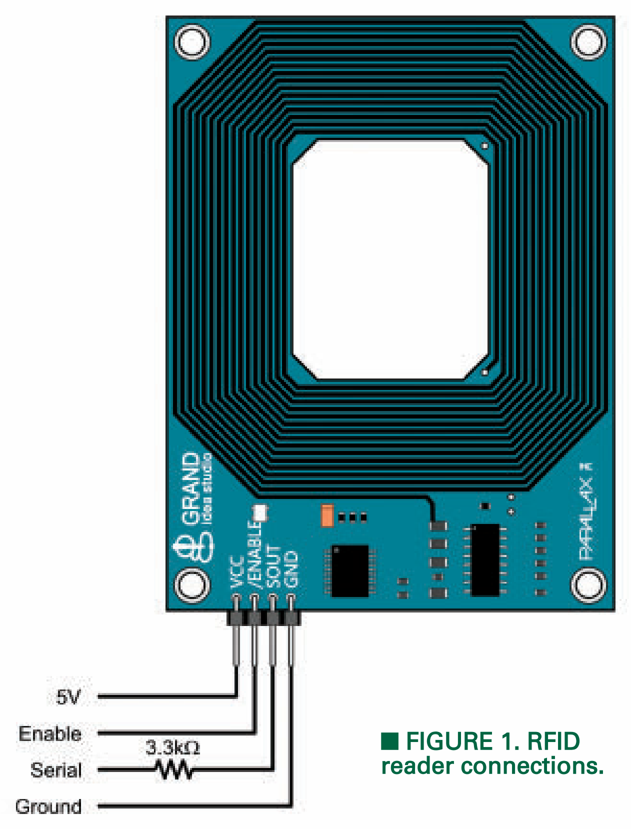


■ FIGURE 2. RFID harness.

Here's where I landed: I created an RFID object that — when called frequently as we did in the LED examples — takes care of activating/deactivating the reader as desired, and provides tag data when a tag has been presented. Let's have a look.

Getting connected to the RFID reader is very simple: It requires a 5V supply, an enable output, and a serial input. The enable pin on the reader is active low and has a pull-up. To activate, we pull this line low; to deactivate, we release the pin to the pull-up.

I confirmed with designer Joe Grand that the serial output is driven. This means we must insert a 3.3K or higher resistor into the serial line as shown in **Figure 1**. My favorite little development tool is the Propeller Activity board (PAB) which includes this resistor on each of its servo headers. To simplify connections to the PAB, I hacked a couple servo extenders as in **Figure 2**.



■ FIGURE 1. RFID reader connections.

Note that the normal servo extender cables are about a foot long; EFX-TEK carries a version that is nearly six feet long should you want to move the reader away from the controller.

The RFID object uses three standard objects to get everything done: ***jm_io*** is used for pin control; ***jm_time*** takes care of timing; and ***jm_fullduplexserial*** (my derivative of FullDuplexSerial) is for the serial connection to the reader.

Jumping into the ***start()*** method, we find that it expects two pins — one for enable, the other for serial input:

```
pub start(en, rx)

  stop

  io.input(en)
  io.input(rx)

  enpin := en

  cog := sin.start(rx, rx, %1100, 2400)

  if (cog)
    enable

  return cog


pub stop

  if (cog)
    cogstop(cog - 1)
    cog := 0

  disable
  bytefill(@tagbuf, 0, 11)
```

Remember that good object design dictates that restarting an object ensures any cogs that had previously been launched are shut down; this is handled with the ***stop()*** method, which also puts the object into disabled state and clears the RFID tag buffer.

The pins used for the reader are cleared in the master cog and the enable pin number is saved for use in other methods. The last big step is to start the serial interface. Again, we're using a full-duplex serial object, but setting it to half-duplex/open-drain mode (safest when using a driven device like the RFID reader) at 2400 baud. If the serial cog loads, the reader is enabled.

The real work is done in the ***check()*** method:

RESOURCES

Jon "JonnyMac" McPhalen
jon@jonmcphalen.com

Parallax, Inc.
Propeller boards, chips, and programming tools
www.parallax.com

Grand Idea Studio
www.grandideastudio.com

```
pub check | c, idx

  if (state == ENABLED)
    if (ina[enpin] == 0)
      c := sin.rxcheck
      if (c == 10)
        bytefill(@tagbuf, 0, 11)
        repeat idx from 0 to 9
          c := sin.rxtime(5)
          if (c => 0)
            tagbuf[idx] := c
        state := READY
        io.input(enpin)

      else
        if (time.millis => ON_MS)
          time.start
          io.input(enpin)

    else { reader off }
      if (time.millis => OFF_MS)
        enable

  return state
```

This method uses the timer object and is designed to be called very frequently — just as we did in the LED examples earlier. When the reader object is enabled, this method looks at the state of the enable pin. If it's low (reader coil is activated), we check the serial input to see if anything is available. If there is and that byte is the RFID header, we drop into a loop that accepts 10 bytes from the serial stream to the the *tagbuf[]* array. The object state is updated to READY and the reader coil is deactivated (to save energy until the current tag is picked up by the application).

Before we go any further, I should explain the serial output of the RFID reader. When a valid tag has been presented, the reader will output 12 bytes, though we only care about the first 11. The first byte will be a decimal 10 ($0A, linefeed). The next 10 bytes will be ASCII characters that are hex digits: "0"..."9", "A"..."F." The final byte in the stream is 13 ($0D, carriage return) which we don't bother with.

Looking at the code, you'll see that *tagbuf[]* is 11 bytes. Before accepting new data from the reader, the buffer is cleared to all zeroes. Since the tag bytes are ASCII, this allows us to treat the array like a z-string and simplifies comparing the received tag to a set of known tags (more on this later).

If no byte was available and the coil on-time period has expired, the reader coil is de-energized and the timer is reset. In operation, you'll see the status LED on the reader alternate between red (ready for a tag) and green (disabled).

In an application, we'll call ***check()*** until the READY status is returned. If we simply wanted to display the RFID tag bytes, we could do that with the ***address()*** method which provides the hub address of *tagbuf[]* — like this:

```
  repeat
    if (rfid.check == rfid#READY)
      term.str(rfid.address)
      term.tx($0D)
      rfid.enable
```

Note that we have to re-enable the RFID object manually; this allows the application to do everything required with the current tag before accepting another.

If we want to associate the tag with other data, we need to build a database of known tags. An easy way to do this is with a DAT table:

```
dat

  Tags
  Tag0      byte      "0415148F26", 0
  Tag1      byte      "0415148E0C", 0
  Tag2      byte      "041514AEA3", 0
  Tag3      byte      "041514A076", 0
  Tag4      byte      "04129C1B43", 0
```

As you can see, we're storing the known tags as ASCII z-strings. Storing the tags in this manner makes searching through the list very simple. The RFID object has a method called ***tag_index()*** that will tell us the index of the new tag in a list of tags known to the application:

```
pub tag_index(p_tags, tmax) | idx

  idx := 0

  repeat while (idx =< tmax)
    if (strcomp(@tagbuf, p_tags))
      return idx
    else
      ++idx
      p_tags += 11

  return -1
```

This method takes a pointer to a list of known tags (e.g., the hub address of our DAT table) and the highest tag index. If we have five tags, our valid index range is zero to four.

We initialize the index to zero and drop into a loop that will run only through the valid index range. If the tag is never found, this loop will terminate and -1 is returned.

Searching is simplified with the ***strcomp()*** function. This requires pointers to two z-strings. If they match, the result is true and the current index is returned. If there is no match, we update the test index and then move forward to the next tag in the database by adding 11 (10 tag bytes plus the terminating 0) to its pointer.

Okay, then. Let's put this to use in an updated demo. To reinforce the use of the timer, we're going to create a little program that displays a running clock which will be the time from start-up or when the last valid tag was presented to the system (unknown tags are ignored). **Figure 3** shows the output when a valid tag is presented.

A very simple loop maintains the running clock display and checks for valid tags. The clock comes from the timer object with includes a ***seconds()*** method:

```
pub show_clock(x, y) | s, m, h

  s := time.seconds

  if (s == lastsecs)
    return

  lastsecs := s
  h := (s  / 3600) // 24
  m := (s // 3600)  / 60
  s //= 60

  timestr[0] := h  / 10 + "0"
  timestr[1] := h // 10 + "0"
  timestr[3] := m  / 10 + "0"
  timestr[4] := m // 10 + "0"
  timestr[6] := s  / 10 + "0"
  timestr[7] := s // 10 + "0"

  cursor_xy(x, y)
  term.str(@timestr)
```

We start by grabbing the seconds value and comparing it to the last time we checked. If they match, we immediately return.

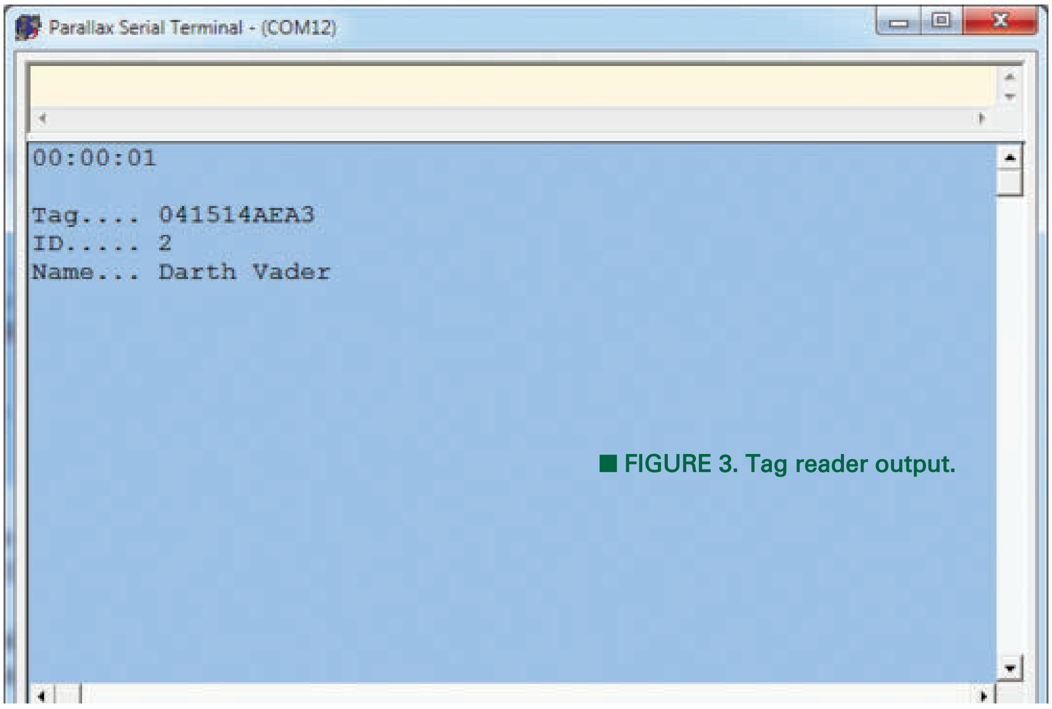
If we have a new seconds value that is copied to *lastsecs* for net time, the seconds value is broken into hours, minutes, and seconds for formatting. The *timestr[]* array is, in fact, a z-string in a DAT block. This allows us to build just the digits as shown, and that string is available to other aspects of the program. In our case, we're going to move the cursor and output the updated time to the terminal. The ***check_tag()*** method does that. It looks to see if the reader has captured a valid tag:

```
pub check_tag | state, tidx, t1

  state := rfid.check

  ifnot (showtag)
    if (state == rfid#READY)
     tidx := rfid.tag_index(@Tags, LAST_TAG)
      if (tidx => 0)
        display_tag(tidx)
        showtag := true
        time.start
        t1 := time.millis
        lastsecs := -1
      else
        rfid.enable

  else
    if ((time.millis - t1) => DISPLAY_MS)
      clear_tag
      showtag := false
      rfid.enable
```

At the top, we call the reader to get its state. If there is not tag information on the screen and the reader has a new tag, we validate that tag with the ***rfid.tag_index()*** method. If it is, in fact, valid, we display the tag information. The timer is restarted and a sync point is created to clean up the display after four seconds. The sync point probably seems redundant and, in fact, it is with this example (since we're resetting the timer when a known tag is presented). If we choose not to reset the timer on tag detection, the sync point is required.

Figure 4 shows my little test setup with the RFID reader connected to the Activity board. For those that want to take things a little further, I've included another example in the downloads at the article link that reads the tags database from an SD card. That version makes use of the parser object from the April column. This allows us to define a tag and give it a name.

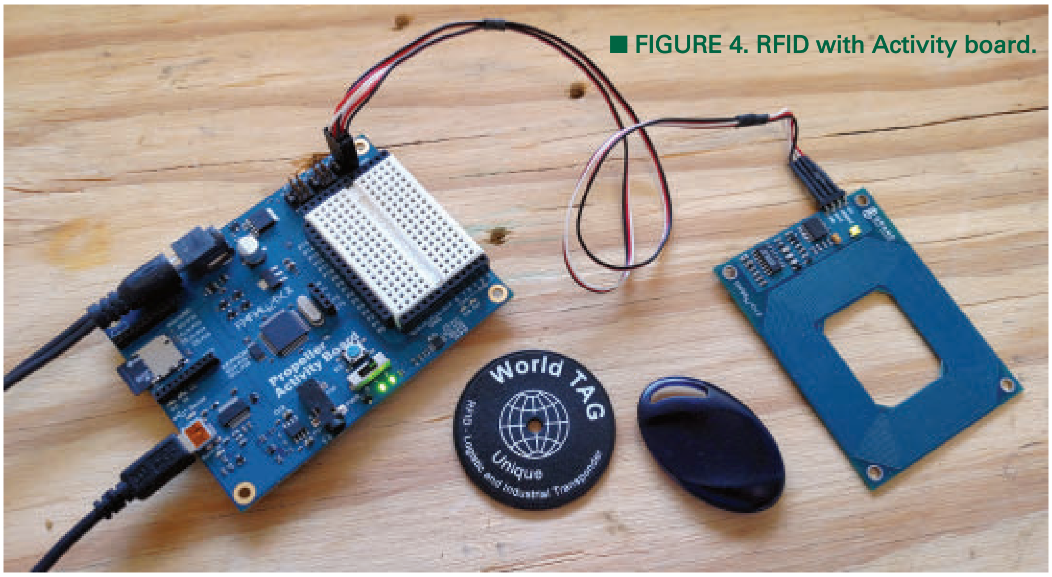
Finally, for those with



```
00:00:01

Tag.... 041514AEA3
ID..... 2
Name... Darth Vader
```

■ FIGURE 3. Tag reader output.

young children, you could take the whole works and add a small powered speaker to it. Build this into a box that your children will enjoy. Now, you can put tags inside of objects and when an object is set on the box, the box can make noise or play music, tell a story, or whatever you think is best to help your child learn. With the microSD socket on the Activity board, you can easily update the tags database and audio to go along with it. Refer to my July 2013 column for playing WAV audio with the Propeller.


■ FIGURE 4. RFID with Activity board.

Okay, then. It's all up to you. If you've avoided playing with RFID because of the high price of tags, you may have missed a big price drop that happened last summer. The pricing fluctuations are explained on the Parallax website. The good news is that you can get an RFID tag for a buck, and spend less when you buy them in quantity.

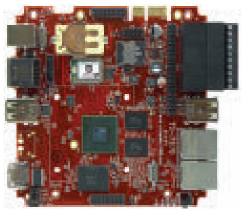
Until next time, keep spinning and winning with the Propeller! **NV**

SUPERIOR EMBEDDED SOLUTIONS

Technologic Systems

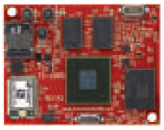
DESIGN YOUR SOLUTION TODAY
CALL (480) 837-5200

www.embeddedARM.com

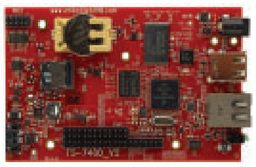


TS-7970
Single Board Computer NEW!
Industrial High Performance i.MX6 Computer with Wireless Connectivity and Dual GbEth

- 1 GHz Solo or Quad Core Freescale i.MX6 ARM CPU
- Up to 2 GB DDR3 RAM
- 4 GB eMMC Flash Storage
- WiFi and Bluetooth Module
- Linux, Android, QNX Support
- Windows OS Coming Soon!


Module Starting At $89 (Qty 100)

Starting at **$169** Qty 100
$214 Qty 1



TS-7400-V2
Single Board Computer
Low Cost Complete Embedded System with Very Low Power Features and Industrial Temps

- 454 MHz Freescale i.MX286 ARM CPU
- 128 or 256 MB DDR2 RAM
- 2 GB eMMC Flash Storage
- 1x microSD Socket
- 0.5 W Low Power Mode
- -40 °C to 85 °C Temp Range

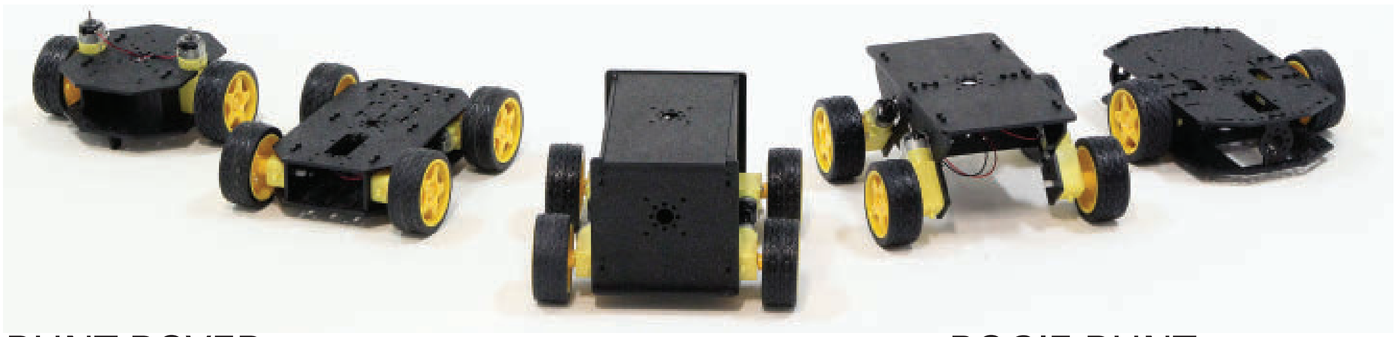
Available with TS-ENC740 enclosure

Starting at **$84** Qty 100
$117 Qty 1

LONG LIFE — We've never discontinued a product in 30 years
RUGGED — Embedded systems that are built to endure
OPEN — Support every step of the way with open source vision
ORIGINAL — Unique embedded solutions add value for our customers

NEW PRODUCTS

- HARDWARE
- SOFTWARE
- GADGETS
- TOOLS



RUNT ROVER ROBOT KITS

A new collection of unique robot kits from ServoCity can spark creativity and imagination, and are fun, affordable, durable, and snap together quickly. The kits can easily be expanded with ServoCity's continually growing Actobotics line.

Whether you're an experienced roboticist or new hobbyist, these kits work for a variety of applications. Simply attach electronics using the innovative multi-board mounts that are compatible with a variety of microcontrollers, such as the Raspberry Pi, Arduino, and the SparkFun Redboard. The Runt Rovers™ are ideal for beginning light programming and educational applications. They are also easily customize with a gripper kit, GoPro mount, phone mounts, sensors, or other components. The platforms offer plenty of space.

You can choose from the Peewee ($15.99); Sprout ($17.99); Junior ($27.99); Whippersnapper ($28.99); or the Half-Pint ($29.99).

BOGIE RUNT ROVER KIT

Bogie is the latest Runt Rover to join theServoCity's new family of robot kits. The Bogie kit is designed with a Rocker-Bogie suspension, making it ideal for climbing over tough obstacles. With nearly 5″ of ground clearance and 5″ of flex, this little bot is simple to drive.

Powerful gearmotors driving six high-traction rubber tires also give the Bogie amazing climbing capabilities. Modifying the Bogie is simple as any component in the Actobotics product line can be attached. The kit can be powered with a 6.0V-7.4V battery pack or re-chargable batteries. Price is $69.99.

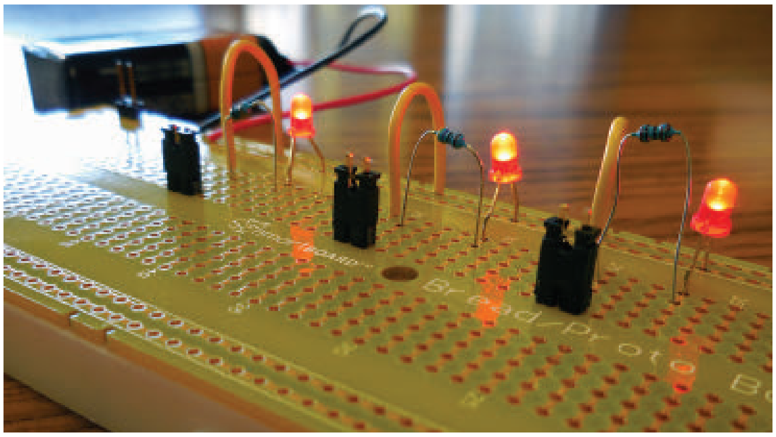
For further information, please contact:

For more information, contact:
ServoCity
Web: **www.servocity.com**

BETTER WAY FROM BREADBOARD to PROTOBOARD

Schmartboard introduces new bread/protoboards. These boards are the exact dimensions of a 400 or 830 tie point breadboard with all of the same traces and power rails.

The act of taking a circuit from a breadboard and transferring it to a protoboard can be frustrating and problematic. With this new board, users can either remove the parts from a standard breadboard



NATIONAL RF, INC.

TYPE 75-NS-3

MINI HF RADIO

The 75-NS-3 covers 3.5 to nearly 11 MHz and is offered as a semi-kit.

CUSTOMER PUTS IT IN A MEAT CAN!!
Visit www.NationalRF.com for this and other Radio Products!
Office: 858-565-1319

Recycling & Remarketing High Technology

WEIRDSTUFF® WAREHOUSE

Software, Computers, Electronics, Equipment, Doohickies

WE BUY/SELL EXCESS & OBSOLETE INVENTORIES
FREE COMPUTER AND ELECTRONIC RECYCLING
GIANT 10,000 SQ. FT. AS-IS SECTION

384 W. Caribbean Dr.
Sunnyvale, CA 94089
Mon-Sat: 9:30-6:00 Sun: 11:00-5:00
(408)743-5650 Store x324
WWW.WEIRDSTUFF.COM

www.boxedkitamps.com

one at a time (while they are next to each other to assure proper placement) and solder them onto the bread/protoboard, or they can lay the new board on top of their own breadboard, place the parts through the bread/protoboard and into their breadboard. Then, when ready, the components can be soldered from the top to the bread/protoboard, before removing it and component leads from their breadboard, thus saving the step of transferring the components from the breadboard to the new Schmartboard (where errors are most commonly made).

The 830 tie point bread/protoboard (bundled with a matching breadboard) is $15. The 400 tie point bread/protoboard is $10.

For more information, contact:
SchmartBoard
Web: **www.schmartboard.com**

NKC electronics

MDE8051 Trainer by Digilent

NKCelectronics.com/MDE8051

Includes the MDE8051 training board, power supply, serial cable

Purchase Orders are accepted from Educational Institutions, US Government and Research Centers

SDP Stock Drive Products
Setting Ideas Into Motion

One-Stop Shop for
Mechatronic Components

EXPLORE
DESIGN
BUY ONLINE

www.sdp-si.com
no minimum requirement

Designatronics INC.

UPDATES TO ATMOSPHERE ONLINE DEV PLATFORM

Anaren, Inc.'s Wireless Connectivity Group has announced that its Anaren Integrated Radio (AIR) module team has introduced a range of 1.1 updates and improvements to its recently introduced Anaren Atmosphere online development platform. Launched in January 2015, Atmosphere uniquely provides embedded product manufacturers with tools to wirelessly connect their products to the Internet, making the IoT revolution accessible via Bluetooth Smart equipped devices. Following are some of the new features and enhancements.

BLE and Embedded Features:

- Data type selection for embedded functions has been added to enhance and compress the transmission time and size over BLE.
- Function type and return selection have been added to reduce transmission-data size, improve efficiency, and allow more developer control.
- GATT Elements have been added to expose the battery state and battery level of a device's single battery or set of batteries.

Developer Tool Features:

- Improved source download process — downloading a user's project source has been split into two downloads

Continued on page 71

By Devlin Gualtieri, Tikalon LLC

BUILD AN ELECTRONIC LOCK

Radio frequency identification (RFID) devices are ubiquitous in electronic controlled-access systems. Although they're considered to be secure devices, they're a little too expensive for hobby applications. So, here's a design for an electronic lock using an inductance coil.

Introduction

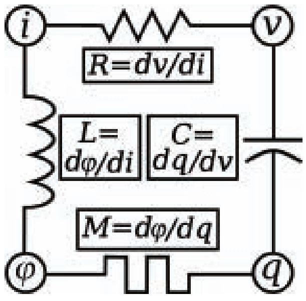
People have been securing their homes and possessions with lock and key devices for millennia. Our electronic age has seen the appearance of non-mechanical locks, such as the wireless entry systems for automobiles. Many of us have employer-issued access cards that serve as door keys, and they also serve as a way to monitor who opens a certain door, and when that happened.

The early access cards used a magnetic stripe. Later, more reliable cards used magnetized wires embedded in the plastic. Presently, RFID devices are used in most access cards. RFID devices are small computer chips that harvest operating power from an electromagnetic field generated by the card reader. They then use this power to transmit a coded signal, sometimes in response to a "challenge" signal. These RFID chips possess a lot of computing power, and high level cryptographic protocols can be employed to make for very secure access. The access cards themselves can be quite inexpensive, but an RFID reader is somewhat costly. In a typical access system, this cost is shared among many access cards, so the cost is warranted for the security provided.

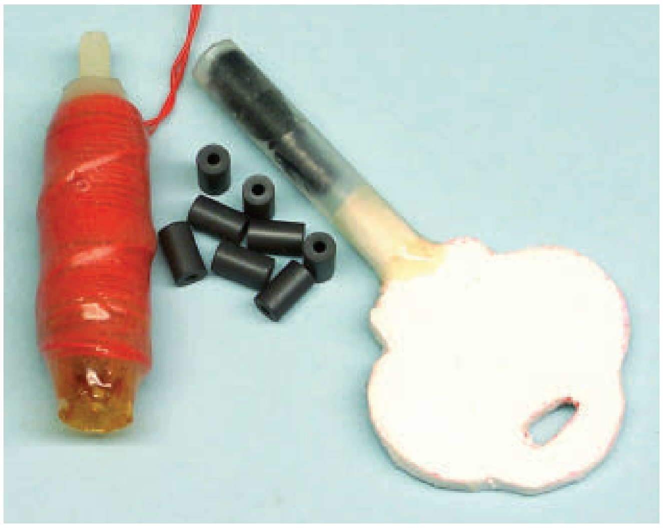
If you're interested in implementing an electronic key for a hobby project, RFID is definitely too costly. Wireless access systems are a little less expensive, but not by much. This article presents a simple and inexpensive electronic key system. It has the possible advantage that the key can be made to actually look like a key.

Another Approach

If we go back to basics, nothing in electronics is more simple than the three principal electrical circuit elements. These are the resistor, the capacitor, and the inductor.



■ **FIGURE 1.** The four fundamental electrical circuit elements: the resistor, capacitor, inductor, and memristor. The resistance (R), capacitance (C), inductance (L), and memristance (M) are defined by derivatives of the voltage (v), current (I), charge (q), and magnetic flux (f).



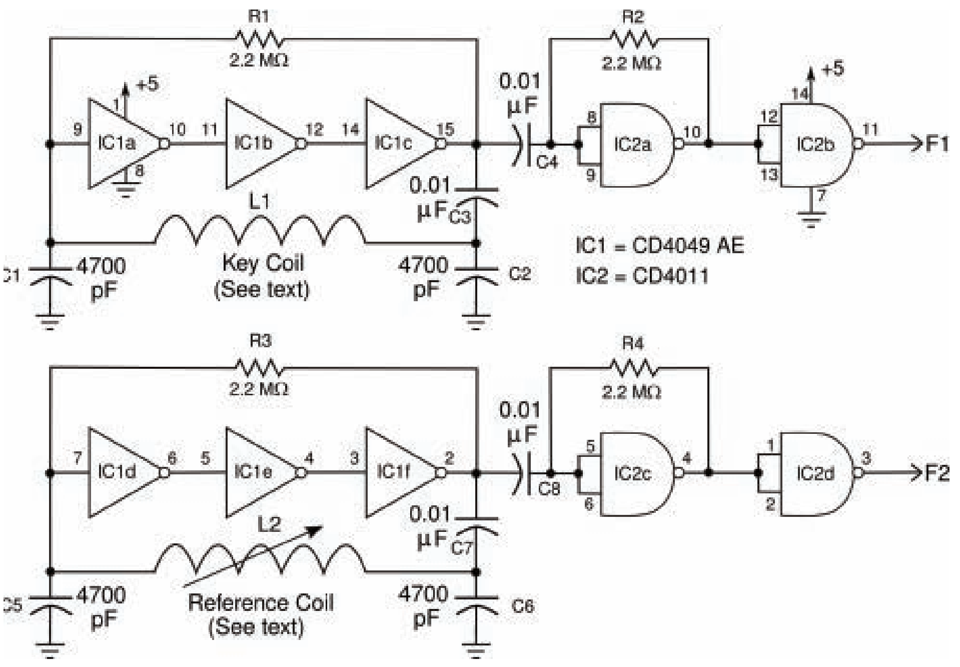
■ **FIGURE 2.** An inductance coil, mating key, and some ferrite core pieces used to make the key. The coil was wound with AWG 38 wire-wrap copper wire, but enameled copper magnet wire could be used.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/june2015_Gualtieri.**

There's a possible fourth circuit element — the memristor — defined by a symmetry in the equations. **Figure 1** shows these four circuit elements.

It would be possible to build an electronic lock in which the key is a particular value of resistance, capacitance, inductance, or memristance. In that case, you would make contact with the key element by two electric terminals to sense the particular value.

It's a lot easier when the key can be read without making electrical contact. Our electronic key is based on an inductance value; but, instead of connecting an inductance, we modify an internal inductance by inserting a ferrite core. The ferrite — which is embedded in the key — increases the inductance by a fixed amount, and the presence of this particular inductance sends an "unlock" signal to a device.



■ **FIGURE 4.** Schematic diagram of the oscillator portion of the electronic key system.

Figure 2 shows an example of an inductance coil and mating key. There's an inch long inductance coil, wound from about 125 turns of insulated wire in three layers, on a 1/4 inch inside diameter cylindrical coil form. The wire used here is 38 AWG plastic-insulated "wire-wrap" copper wire, but enamel-coated "magnet" wire could have been used instead. The key is a mating plastic tube in which small ferrite cores of the sort used to make simple radio frequency (RF) chokes are glued inside. Inserting the key into the inductor changes the inductance by a particular value. A greater or lesser number of ferrite core pieces can be used to make different key values.

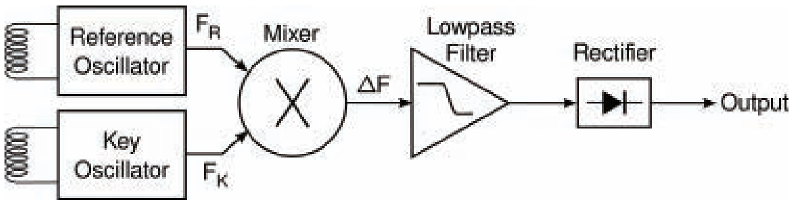
Basic Inductance Lock and Key Circuit

The easiest way to read an inductance is by making it part of an inductance-capacitance resonator in an oscillator circuit. The frequency of this oscillator can be compared to that of a reference oscillator, and an unlock signal can be generated when the frequencies match within a predetermined range.

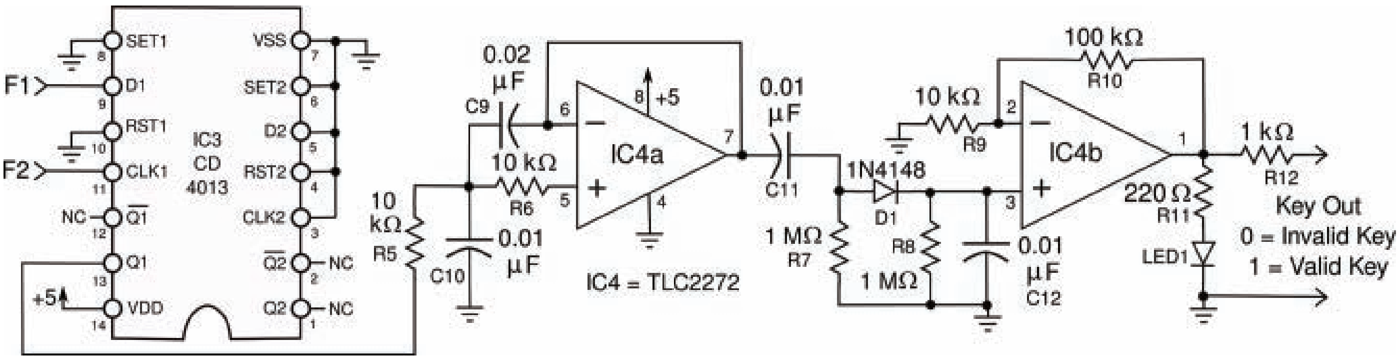
One way to compare frequency signals is to use a mixer circuit. A mixer will produce the sum of its two input frequencies, or their difference. When the frequencies are very close, the difference signal will fall into the audio frequency band where it can be selected by a low pass filter, and then rectified to obtain a control signal. A block diagram of this process is shown in **Figure 3**.

A practical implementation of this circuit is shown in the schematic diagrams in **Figures 4** and **5**. As can be seen from **Figure 4**, it's possible to use resistors to bias inverters of a CMOS logic circuit to function as amplifiers in an oscillator. The gain of each individual inverter of the CD4049 is only 30, but cascading three inverters gives you an amplifier with a gain of more than 10,000. This is sufficient to drive the LC resonator into oscillation. The additional stages of the CD4011 buffer the oscillator signal.

A wide variety of coils can serve as the inductance. I've used anywhere from 100-250 turns of AWG 38 wire in 2-4 layers, wound to an inch length on a coil form with a 1/4 inch inside diameter. These coils oscillate in the range of about 400-1,000 kHz, so you could actually verify operation of most of these by holding an AM radio nearby. The coil shown in **Figure 2** has three layers



■ **FIGURE 3.** Block diagram of an electronic key system using two oscillators. The frequency of the reference oscillator is adjusted to put the frequency difference between it and the key oscillator into the bandpass of the filter.



■ **FIGURE 5.** Schematic diagram of the mixer, low pass filter, and rectifier of the electronic key.

of 35 turns each.

Using that coil, the oscillator had a frequency of 516 kHz when no key was inserted. Inserting a key containing two ferrite cores decreased this to 387 kHz. A key with three ferrite cores caused oscillation at 286 kHz.

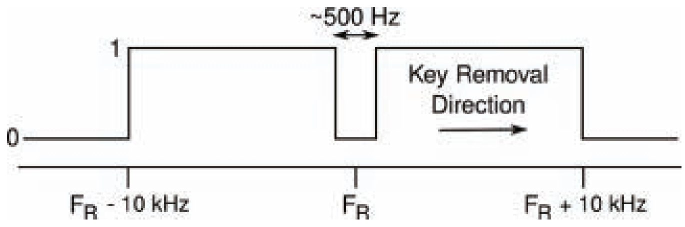
The easiest way to make the reference coil is to wind a coil identical to the key coil, and then attach an assembly that allows a controlled insertion of a tube filled with ferrite. In my case, I used a nylon bolt and nut for this purpose since conductive pieces (such as metal) are not allowed.

The schematic diagram in **Figure 5** shows the mixer, low pass filter, and rectifier. The mixer uses a CD4013 "D" type flip-flop as a digital mixer. A precaution observed in using CMOS logic is to ground unused inputs since these might float "high" and draw excessive device current. Such digital mixers do produce a difference signal, but they also respond to harmonics of the input frequencies. This "feature" doesn't matter in our circuit.

The low pass filter is a two-pole Butterworth with a corner frequency of about 2 kHz. This circuit is characterized by one capacitor (C9) being twice as large as the other (C10). The easiest way to implement this is to just buy three capacitors of the lower value, and make the higher value using two parallel capacitors. This makes a lot of sense in this circuit since there are many 0.01 µF capacitors.

The choice of operational amplifier is not critical. You can use any "rail-to-rail" type operable on a five volt supply. Rectification is accomplished with diode D1, the rectified signal is filtered by C12, and then amplified. There's a logic level output you can use to drive other circuitry; for example, an optically-isolated solid-state relay, or just a power transistor for driving a solenoid.

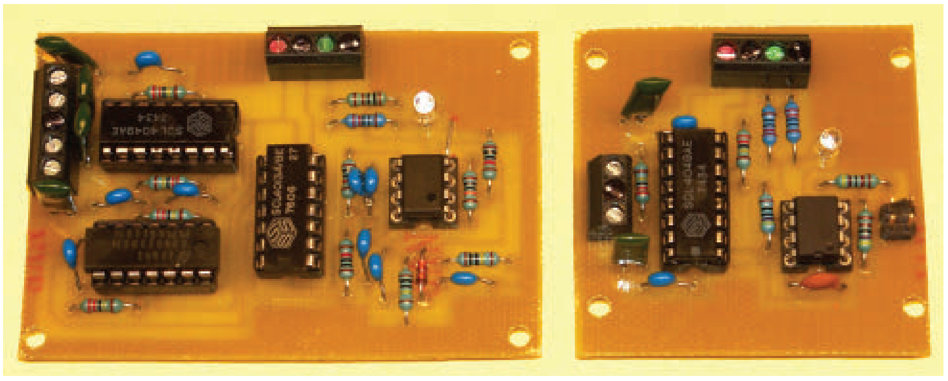
The circuit is calibrated by inserting the key and then adjusting the reference inductor to give a slightly lower frequency within the passband of the low pass filter. This setting is aided by the LED, which lights when the frequency condition is matched. Note that setting the frequencies exactly equal is not desirable since the low difference frequency will not filter well.



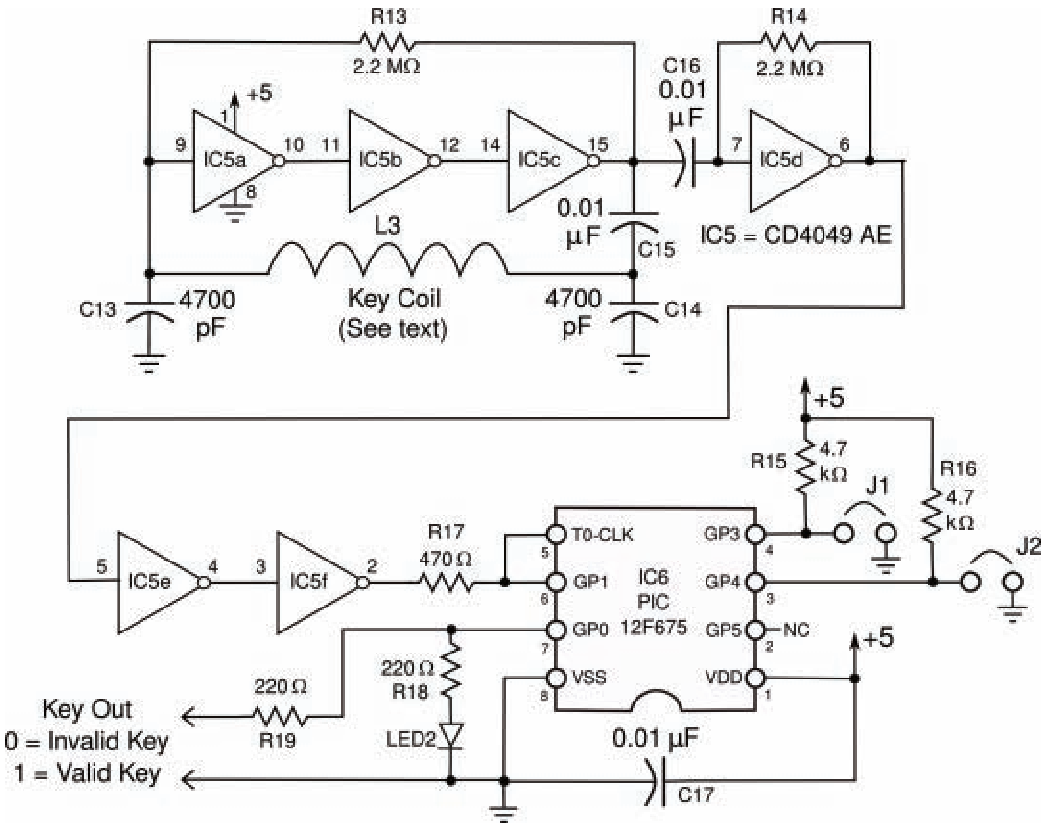
■ **FIGURE 6.** Performance of the mixer circuit. The LED will light when the key oscillator frequency is within a range of the reference oscillator frequency.

As shown in **Figure 6**, the mixer responds to frequency differences on both sides of an exact frequency match. That means you can also set the reference oscillator for a higher frequency than the key oscillator. This will work, too, but there will be a blip in the output as the key is inserted or extracted as it passes through the zero point.

The circuit shouldn't generate any radio frequency interference; but, as a precaution, it should be built inside a grounded metal enclosure. You can ensure that radio frequency interference is not being emitted by placing



■ **FIGURE 8.** Photos of the circuit boards of the two versions of the electronic key circuit. The microcontroller version (right) uses fewer components, but it does require a programmed microcontroller chip.



■ FIGURE 7. Schematic diagram of the microcontroller version of the electronic key system.

an AM radio a few feet away and scanning the band. As mentioned earlier, the coils will resonate at AM radio frequencies. Or, if they resonate below, their second harmonic signal will be in the AM radio band.

Simplify with a Microcontroller

This circuit functions as expected, and it's built from commonly available components. I had all the parts in my shop drawers, and the CMOS ICs were left over from some very old projects. The only problem the circuit has is that a lot of components are required for such a simple task. One way to decrease component count in a circuit is to throw some software into the solution. For that reason, I've designed another electronic lock system using a microcontroller. The schematic for this is shown in **Figure 7**.

As you can see in **Figure 7**, we have the same sort of oscillator as in the first circuit, but there's only one. Its output goes to the microcontroller, which acts as a frequency counter. The frequency is compared against stored values that correspond to the presence of up to three different keys.

To measure the frequency, we merely count the number of pulses that occur in a set time interval. To do this, we need a timer and a counter. The PIC microcontroller has a 16-bit timer, Timer 1, that allows us to set a fairly accurate period using the internal clock oscillator. In our case, we count pulses in 40 millisecond intervals.

Getting an accurate pulse count presents a small problem since the remaining counter is only eight bits, which would allow reading the frequency to one part in 256. We're saved by a trick — published on the microcontroller manufacturer's website — that allows 16-bit resolution by some software trickery involving resistor R17 and a spare microcontroller pin.

Since the counter input passes through an eight-bit prescaler, we use the spare pin to step additional pulses into the prescaler. By counting how many pulses are needed to increment the counter, we can calculate the prescaler count, thereby adding its eight-bit resolution to our existing eight-bit counter. In this way, we get 16-bit resolution of the frequency.

The microcontroller is calibrated to up to three different keys using the jumpers, J1 and J2. In normal operation, J1 and J2 are left open. If one or both jumpers is sensed when power is supplied and the microcontroller program reboots, the software stores whatever frequency value it finds. Having a key present when that happens stores the key value for comparison in normal operation.

After the key value is stored, the program halts. You then remove the jumper (or jumpers) and the circuit will function as a lock matched to that key on subsequent power-ups. As shown in **Table 1**, up to three keys can be stored, so one of these keys can be a "master" key. In this sense, the microcontroller circuit has another advantage other than just needing fewer components.

■ Table 1

State	J1	J2
Operate	OUT	OUT
Key 1	IN	OUT
Key 2	OUT	IN
Key 3	IN	IN

Software

The source code — written in PIC BASIC PRO — is available at the article link, along with hex files for using the code without a compiler. The software also serves as a tutorial on how to use a PIC timer as a frequency counter. You still need to program the microcontroller, but there are quite a few circuits for inexpensive programmers on the Internet, and also a few inexpensive programmers, themselves.

Caveats and Conclusion

A few caveats are in order. This system is about as secure as the simple lock and key devices in furniture and children's toys. It's not as secure as most RFID locks, and I certainly wouldn't secure my house with one. It would be good for locking-out TVs and game systems during homework time, or as a lock for a child's lock-box or diary.

Of course, as with any electronic lock system, you should have an alternative means of opening your lock in the event of a circuit or power failure. The paper clip inserted in a hole that once was needed for computer optical discs is one possibility.

Also, you needn't be limited to the key shape presented. You can have a card instead of a key, with a flat coil to accept the card and some ferrite powder glued between sheets of plastic. If you decide to crush ferrite cores to make a powder, remember to wear safety glasses! Wrapping them in paper and crushing in a vice is a typical approach. **NV**

References

The datasheet for the PIC 16F675 is available from Microchip at **ww1.microchip.com/downloads/en/devicedoc/41190c.pdf**

Stan D'Souza, "Frequency Counter Using PIC16C5X ," Microchip Technology, Inc., Application Note AN592 at **ww1.microchip.com/downloads/en/AppNotes/00592d.pdf**

PIC BASIC PRO is available from M.E. Labs, Inc.
2845 Ore Mill Road, Ste. 4, Colorado Springs, CO 80904
phone 719-520-5323; **www.melabs.com**.

PARTS LIST

ITEM	DESCRIPTION	ITEM	DESCRIPTION
Circuit with Mixer		*Circuit with Microcontroller*	
R1-R4	2.2 megaohm	**R13-R14**	2.2 megaohm
R5-R6, R9	10K ohm	**R16R15-**	4.7K ohm
R7-R8	1 megaohm	**R17**	470 ohm
R10	100K ohm	**R18-R19**	220 ohm
R11	220 ohm		
R12	1K ohm	**C13-C14**	4700 pF NPO
		C15-C17	0.01 µF
C1-C2, C5-C6	4700 pF NPO		
C3-C8, C10-C12	0.01 µF	**L3**	Key Coil (Note 2)
C9	0.02 µF (Note 1)		
		IC5	CD4049 AE
D1	1N4148	**IC6**	PIC12F675
L1	Key Coil (Note 2)	**LED2**	Red LED
L2	Reference Coil (Note 2)		
		J1	Jumper
IC1	CD4049 AE	**J2**	Jumper
IC2	CD4011		
IC3	CD4013		
IC4	TLC2272 (Note 3)		
LED1	Red LED		

Misc
Five volt Power Supply
IC Sockets
Terminals
Hardware
PCB *(Patterns are available at the article link)*

Notes:
(1) Can be a parallel combination of two 0.01 µF capacitors.
(2) See text.
(3) Most five volt rail-to-rail operational amplifiers will work. You need two, or a dual chip.

AUTHOR BIO

Dev Gualtieri received his Ph.D. in Solid State Science and Technology from Syracuse University in 1974. He had a 30 year career in research and technology at a major aerospace company and is now retired. Dr. Gualtieri writes a science and technology blog at **www.tikalon.com/blog/blog.php**. He is the author of two science fiction novels, and books about science and mathematics. See **www.tikalonpress.com** for details.

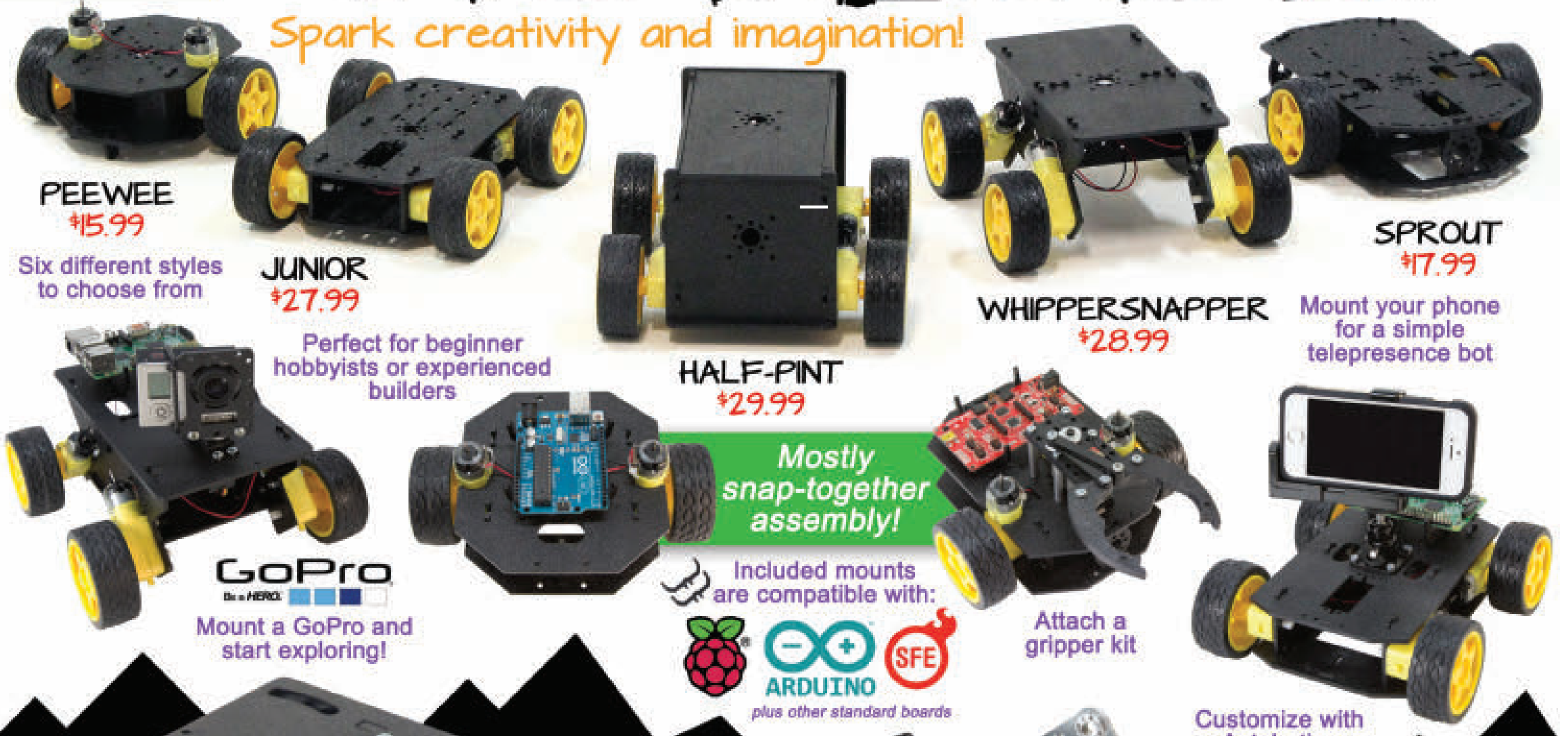
NEW! Actobotics™

DREAM. DESIGN. BUILD. *REPEAT.*

sales@servocity.com
620.221.0123

RUNT ROVERS™

Spark creativity and imagination!



PEEWEE
$15.99

Six different styles to choose from

JUNIOR
$27.99

Perfect for beginner hobbyists or experienced builders

HALF-PINT
$29.99

WHIPPERSNAPPER
$28.99

SPROUT
$17.99

Mount your phone for a simple telepresence bot

Mostly snap-together assembly!

Included mounts are compatible with:

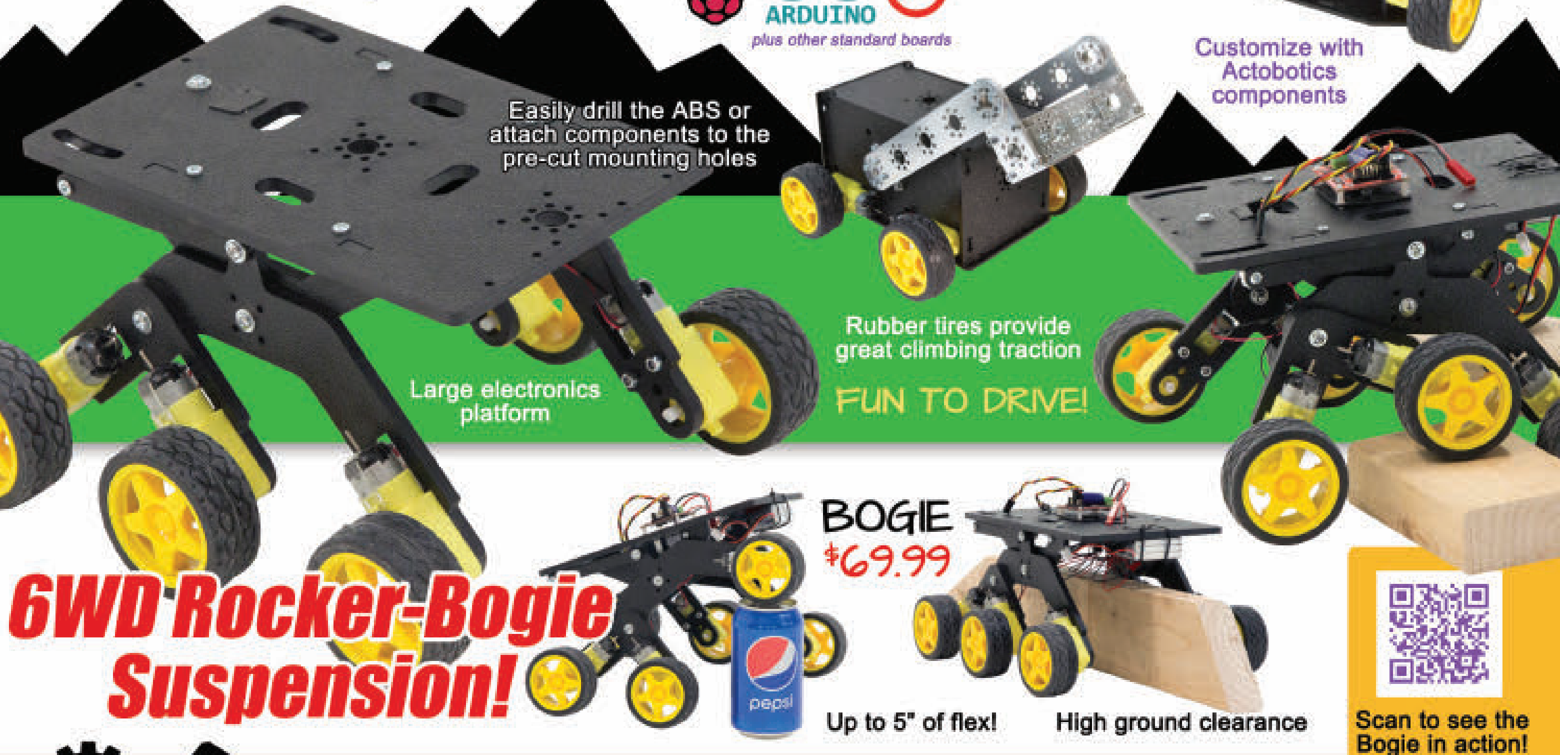
ARDUINO SFE

plus other standard boards

Mount a GoPro and start exploring!

Attach a gripper kit

Customize with Actobotics components



Easily drill the ABS or attach components to the pre-cut mounting holes

Large electronics platform

Rubber tires provide great climbing traction

FUN TO DRIVE!

BOGIE
$69.99

6WD Rocker-Bogie Suspension!

Up to 5" of flex!

High ground clearance

Scan to see the Bogie in action!

www.SERVOCITY.com

Great educational discounts available. Apply today!

Fun contests with awesome prizes - like $1,000 worth of parts!

By Ron Newton

BUILD A WIRELESS FREEZER ALARM

If you've ever had a freezer malfunction with hundreds of dollars' worth of meat rotting in it, you know what a disaster it can be. Not only is there the expense of the loss of the contents, but the smell and the cleaning is something you will never forget. Many of us have freezers in sheds, barns, and external garages. We never would know until the smell waifs through our nostrils if the power or compressor goes out.

This is a wireless device that performs two functions. It alarms if the power is disconnected or a circuit breaker trips, and it constantly measures the internal temperature of the freezer. It will sound the alarm if the temperature exceeds -15°C (or about 0°F). The temperature can be set to alarm at any temperature you require. The Linx transmitter is capable of transmitting up to 3,000 feet, depending on what is in its path.

Circuitry

TRANSMITTER

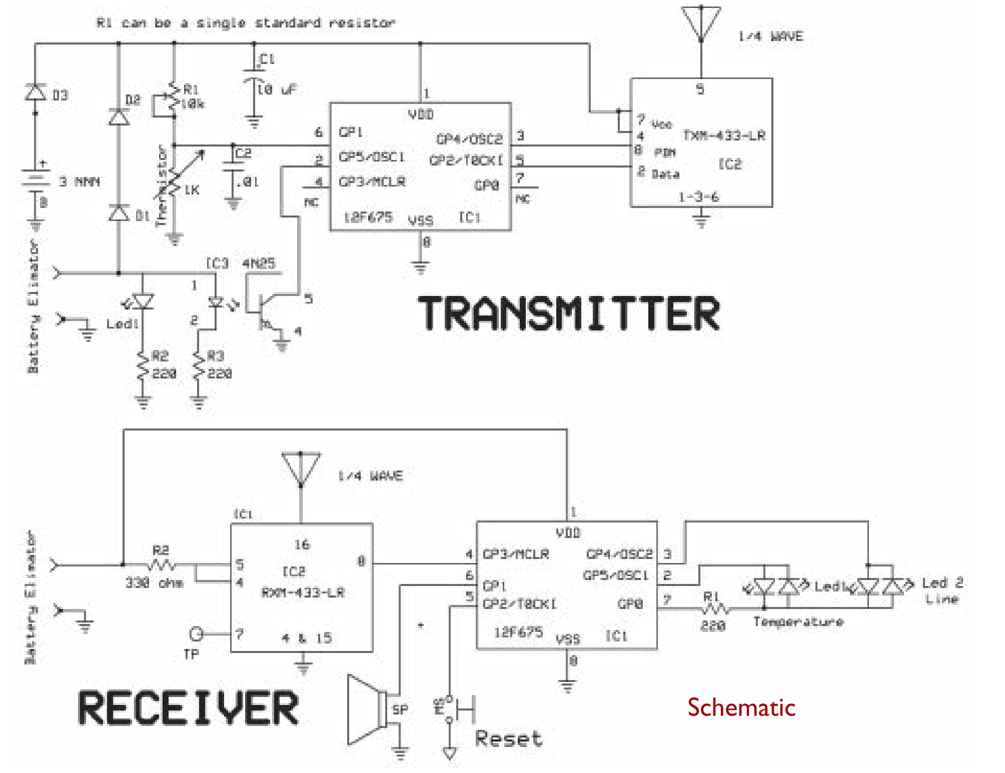
The transmitter unit uses a Linx TXM 418 long range transmitter and a Microchip PIC12F675 for control. This chip has an analog-to-digital converter (ADC). The chip also provides 28 transmission codes, so if there are several transmitters in the area, they won't trip other people's alarms. ***The assembly code for both the receiver and transmitter, ExpressPCB board files, schematics, parts source, and other Hints and Tips are available at the article link.*** If you don't have a programmer, preprogrammed chips are available in the *N&V* webstore

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/june2015_Newton**.

along with boards.

For power, our project uses three N batteries for transmission when the power goes out. The batteries are isolated from the power supply by blocking diode, D3. The battery should last the shelf life of an N battery since the batteries are only used if there is a power outage. The unit draws its power from the five volt battery eliminator when the power is on. Power from the battery eliminator is dropped by using two diodes (D1-D2) in series.

The power outage detector uses a NPN optoisolator, which pulls GP1 (with an internal pull-up resistor) of the micro low when the power is on. When the power goes off, the GP1 goes high signaling the transmitter to indicate a failure. The power also goes to an indicating LED which shows the unit is plugged in.



Schematic

The temperature sensing uses a system that I have used in Africa for many years for blood bank refrigerators and plasma freezers. The recommended temperature for freezing food is below -15°C. The unit uses a very small 1K thermistor (at 25°C) in series with a 10K resistor. This forms a voltage divider.

Thermistors are not linear devices, and follow a complex formula. The thermistors resistance increases as its temperature drops. I have used the ADC of the PIC12675 to detect the tripping point. I have set this ADC to trip at a ratio of .5 of the Vcc. This will allow people who don't have programmers to set the temperature of the alarm using a 10K potentiometer or a resistor. I recommend that you check the temperature of your freezer first in Centigrade to determine where the alarm should be set.

I checked my refrigerator/freezer in the garage and the lowest I could set it was -10°C, but then the main part of the refrigerator was freezing. I ended up setting it to -3°C. Most chest freezers will go down to -15°C without a problem.

Once you check your freezer temperature, open the Excel spreadsheet titled "Voltage Ref" at the article link. Either set your potentiometer (R1) by looking up the resistance of the thermistor, or you can use a standard resistor of the same value since the thermistor would be at the given temperature. The reason I use a ratio instead of a voltage is that the ADC uses Vcc as a reference. The thermistor and R1 are across Vcc. The ratio remains constant even though the voltage of Vcc may change. The trip point of the alarm is set to .5

Here is an example:

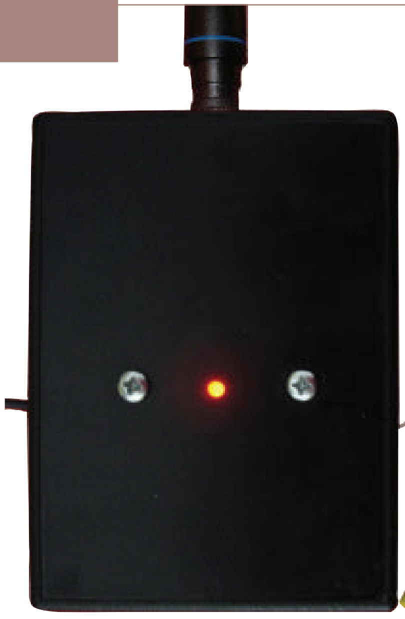
Let's say that your freezer is running around -15°C. I would set the alarm to –10°C so that when you open the door, it won't set off the alarm. Using the lookup table at the end of this article, the resistance of the thermistor at -10°C is 4,569 ohms. If you use a standard 4.7K resistor, you would alarm between -10°C and -11°C. You can also use the 10K pot and tweak it for this application.

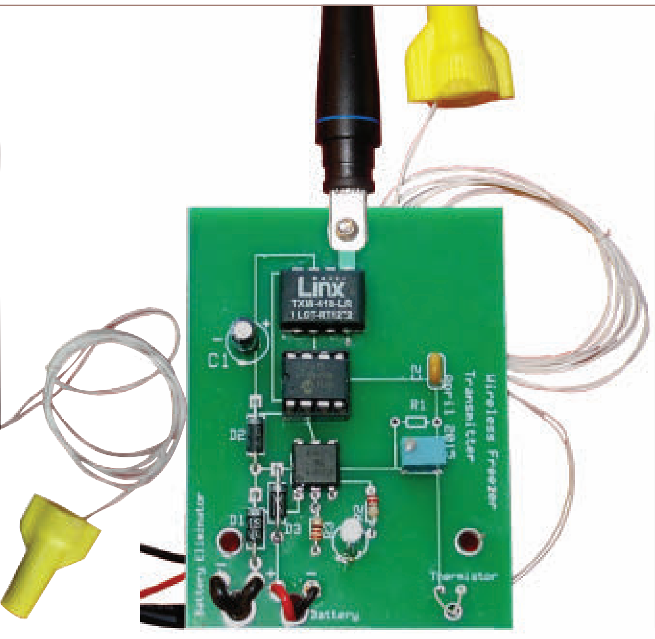
The voltage is constantly being monitored by the micro; when the voltage of the voltage divider has a ratio below .5, it turns on the alarm.

For transmitting data, I used a modified RS-232 bit-banging program. Two bytes (eight bits each) of data are sent to the receiver for translation. The first byte 28 has 256 codes. The second byte is used for data transmission. Only the first two bits are used. If you are going to change the code for the transmitter and the receiver (requires a programmer), they have to be identical and are set in the software, e.g., the first byte can be 00 to FF hexadecimal.

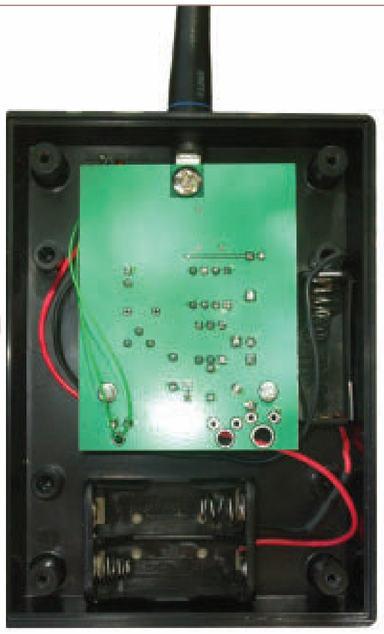
RECEIVER

The receiver is powered by another five volt battery eliminator and can be wall mounted. It uses a Linx TRM 418 long range receiver and a 1/4 whip antenna. The micro used is a PIC12F675; it is a small workhorse that is quite inexpensive.


■ **FIGURE 1.** Transmitter box.



■ **FIGURE 2.** Transmitter board.


■ **FIGURE 3.** Battery holders.

It has two bi-colored LEDs that change from red to green, depending on how the two leads are powered from five volts to ground. Reversing the LEDs causes the colors to change. They normally should be green.

If the receiver receives an alarm transmission, it turns on the piezo buzzer which squeals. This can be silenced by pushing a reset switch.

As with most projects, there are obstacles. This one was no exception. I had ordered boards, started programming, and was going to use a multiplexing scheme

WORLD'S MOST VERSATILE CIRCUIT BOARD HOLDERS

Our Circuit Board Holders add versatility & precision to your DIY electronics project. Solder, assemble & organize with ease.

VISIT US ON

MONTHLY CONTEST

Visit us on Facebook® to post a photo of your creative PanaVise project for a chance to win a PanaVise prize package.

PANAVISE®

Innovative Holding Solutions

7540 Colbert Drive • Reno • Nevada 89511 | (800) 759-7535 | www.PanaVise.com

AP CIRCUITS

PCB Fabrication Since 1984

As low as...

$9.95

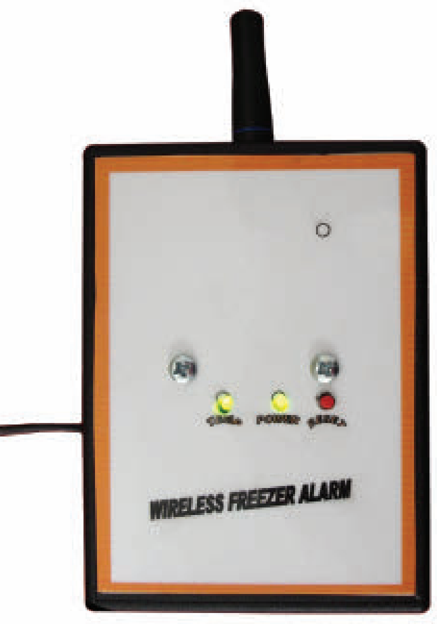
each!

Two Boards
Two Layers
Two Masks
One Legend

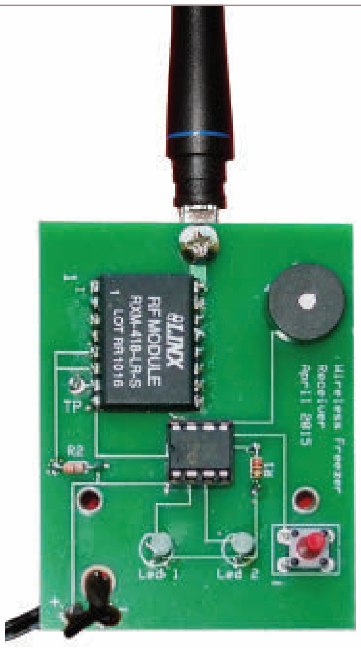
Unmasked boards ship next day!

www.apcircuits.com

VISA MasterCard PayPal IPC MEMBER ASSOCIATION CONNECTING ELECTRONICS INDUSTRIES BBB



■ **FIGURE 4.** Receiver box.



■ **FIGURE 5.** Receiver board.

for driving the two LEDs. Unfortunately, the multiplexing caused timing problems when detecting the RS-232 sequence, and I had used all my ports on the PIC12F675. I thought I would have to redesign the board using a 16-pin micro. I solved the problem by a sequence of software codes. If both LEDs are red or green, I used a common ground. When one LED is green and the other is red, I turn off the common and use the LEDs to drive each other.

Linx's layout is extremely critical and you must use a ground plane. Don't try to breadboard. It won't work. If you modify the board layout for another project, make sure that you don't have any components within .15" of the chip, and don't ever run any traces under the chip.

When the receiver detects a start byte, it takes the first byte and compares it to a known code. If this passes, the first two bits of the second byte are checked to see what alarm caused the interruption and it's deciphered. This activates the LEDs and sounds the alarm. I normally use the micro to drive the transducer. However, this again interfered with the RS-232 sequencing. I wanted both the LEDs and sound to be on when the receiver is checking for a signal.

Building the Units

TEMPERATURE SENSOR

Make a cord for the thermistor by twisting two each of six foot lengths of wire-wrap wire. Strip about 1/2" of the wire-wrap wires and wrap them around the leads of the thermistor. Solder and cut the excess leads off of the thermistor. Take the #12 wire nut and take out the metal spring. Place the thermistor into the wire nut and fill it with hot glue. This will make a waterproof sensor. Since the thermistor is high resistance, the wire can be any length and will not affect the readings.

TRANSMITTER

The transmitter box mounts on the wall using a Serpac box **(Figure 1).** The box comes with all the mounting hardware. There is a template for drilling the holes at the article link. Cut it out and paste it onto the box using a glue stick. You will be able to remove it with hot water. The drill sizes are located on the template. You will need to drill two additional holes for the power cord and the thermistor.

I will leave the location up to you, as it will vary with the mounting of the unit. Consider drilling the hole close to the back, and file it out to the edge. The back cover has a lip, so give some distance from the edge. The reason for doing this is so you can remove the circuit board and the wires without having to de-solder the wires.

Mount the Linx transmitter first **(Figure 2).** Make sure the chip is centered on pads with pin 1 next to the 1, heat the pad, and add solder. Solder in IC2 and IC3, noting pin 1 is the square pad. Solder the three diodes noting their polarity, the three resistors, and the two capacitors noting polarity on C1.

Solder the three N battery holders in series (black to red). Use double-sided tape and place the holders into the box **(Figure 3).**

Cut the plug off the end of the battery eliminator, thread it through the strain relief hole, and solder the white striped wire to the negative and the other to the positive; then solder. Perform the same operation with the battery holders with the red to the positive and black to the negative. Repeat once more with the two thermistor wires. They have no polarity.

Place the LED into the board with its long lead to the square pad. **DO NOT SOLDER**.

Take two each 1/2" 6-32 screws and push them through the top side of the box. Add the two spacers and screw them into the top of the board. The board is designed to act as the nut. Place the antenna into the box with its flat toward the back and secure with the screw provided. Push the LED though its hole and now solder. Secure the N battery holders with double-sided tape. Add the three N batteries and secure the back of the box with the hardware provided.

RECEIVER

Using the template, drill the box holes. Again, you will have to make a hole for the power cord **(Figure 4).**

Solder the Linx receiver chip to the board. Solder the sound transducer noting its polarity, and the resistor. Solder in IC1 and the switch. Cut the plug off of the battery eliminator and thread it through the hole in the box and the strain relief hole. Solder the white wire to negative and the other to positive **(Figure 5).** Place the two bi-colored LEDs into their holes on the board. The long lead goes to the square pad. **DO NOT SOLDER**.

Place two 1/2" 6-32 screws into the box and add the two spacers. Attach the screws to the board. Add the antenna as on the transmitter board. Push the LEDs through the holes in the box so they are visible from the front of the box and solder. Secure the back of the box with the hardware provided.

A mounting bracket is supplied with the boxes for mounting on the wall. Use the screws supplied.

Using the Units

Plug in the receiver box. The LEDs should light and both should be green. Plug in the transmitter. The freezer LED should be red and the power LED should be green; it should alarm. Push the reset button to silence it.

Place the temperature sensor into the freezer. You can add hysteresis by placing a solution of four ounces of 50% glycerin (available at your local drug store) into the freezer. By doing this, the freezer will not alarm if you open the door. **Do not use ethylene glycol** in a food freezer as it is poisonous.

When the sensor drops below its preset alarm, it will transmit to the receiver, then the receiver temperature LED will change from red to green. The transmitter only transmits when there is a change in the alarm situation.

Baby, it's cold in there, but if it's not, you'll know. **NV**

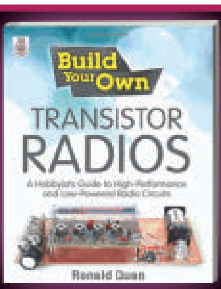
PARTS LIST

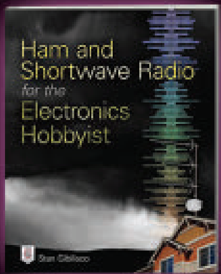
ITEM	DESCRIPTION	QTY	SOURCE
TRANSMITTER			
Antenna	ANT-418-PW-LP1	1 ea	Linx
Battery eliminator	5 volt 100 mA	1 ea	
Battery holder	N	3 ea	
Box	031WI,BK	1 ea	Serpac
C1	10 µF 16V radial	1 ea	
C2	.01 µF 50V	1 ea	
D1 D2 D3	1N4001	3 ea	
IC1	PIC12F675	1 ea	Microchip
IC2	TXM-418 LR	1 ea	Linx
IC3	Optoisolator 4N25	1 ea	
LED	Red 3 mm	1 ea	
R1	10K 25 turn	1 ea	Can be standard resistor
R2 R3	220 ohm 1/8 watt	2 ea	
Thermistor	1K	1 ea	Vishay NTCLE100E3102JB0
RECEIVER			
Antenna	ANT-418-PW-LP	1 ea	Linx
Battery eliminator	5 volt 100 mA	1 ea	
Box	031WI,BK	1 ea	Serpac
IC1	PIC12F675	1 ea	Microchip
IC2	RXM-418-LR	1 ea	Linx
R1	220 ohm 1/8 watt	1 ea	
D2	330 ohm 1/8 watt	1 ea	
Speaker	CX-0905C	1 ea	CUI, Inc.
S1	Switch mom 6 mm H	1 ea	
LED 1-2	3 mm bi-colored	2 ea	
HARDWARE			
Spacer	#6 3/8"		
Screws	6-32 1/2"		
Wire nut	#12		

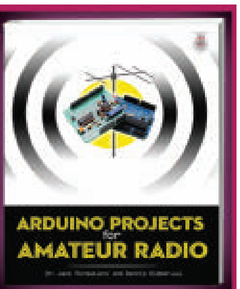
Lookup Table Reference

Temp	Ohms	Temp	Ohms	Temp	Ohms	Temp	Ohms
-40	23342	-35	17336	-30	13018	-25	9877
-20	7569	-15	5855	-10	4569	-5	3596
0	2854	Thermistor Mouser # 594-2381-640-63102					
5	2282	10	1838	15	1491	20	1217
25	1000	30	826.6	35	687.3	40	574.6

DO YOU LOVE RADIOS?







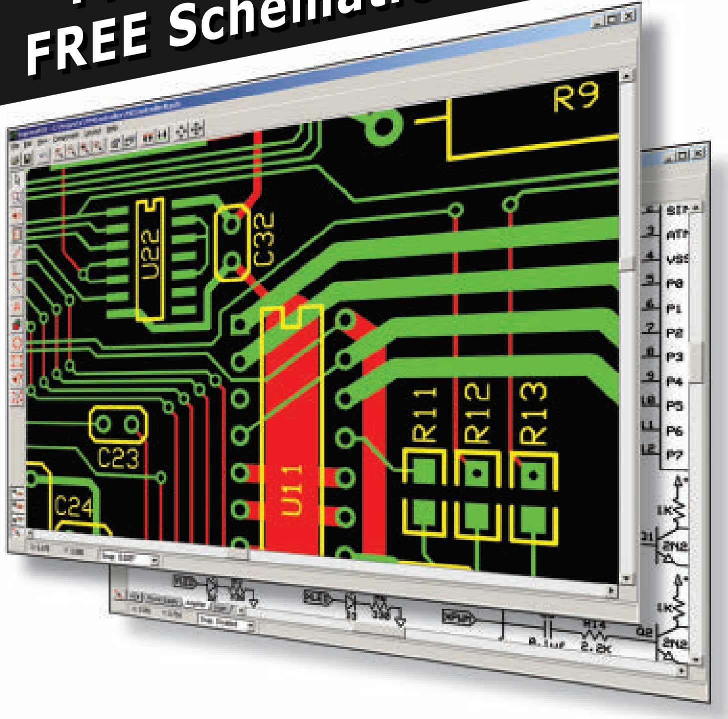
Get 20% off these three books PLUS Free Std. Shipping!

To order call 800 783-4624 or visit: http://store.nutsvolts.com

Limited Time Only

$51 For 3 PCBs

FREE Layout Software!
FREE Schematic Software!



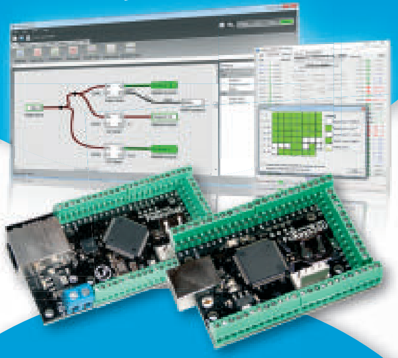
01	DOWNLOAD our free CAD software
02	DESIGN your two or four layer PC board
03	SEND us your design with just a click
04	RECEIVE top quality boards in just days

expresspcb.com

Imagine this...

- a small and inexpensive
- USB or ethernet device
- with a rich set of features
- including PLC functionality and 3-axis stepper motor controller
- all accessible via free .NET, ActiveX or C++ library
- cross-platform
- configurable with free software

PoKeys



Or this...

- all in one: Oscilloscope, Data Recorder, Logic analyzer, Analog and digital signal generator
- smallest USB 2.0 portable 1MS/s oscilloscope
- data acquisition of analog and digital signals
- data recording
- export to CSV, XLS, PDF and HTML
- simple usage of advaced features
- examples for C++, VB, Delphi and LabView
- free software and updates

PoScope Mega1+

PoLabs

Visit us at www.poscope.com

ARDUINO Based Data Acquisition

Quick and easy data acquisition and display with an Arduino

By Eric Bogatin
eric@EricBogatin.com

I love data. Measuring things, plotting the results in a way to instantly visualize the behavior, and — most importantly — analyzing the results. Maybe it's because of my physics training, but even as old as I am, I still get a thrill when I can measure something and have it match the predictions of a simple model. This is especially exciting when I can collect the measurements by computer and utilize the power of easy-to-use yet powerful tools to perform the plotting and analysis.

Why an Arduino Based Data Acquisition System

Figure 1 is an example of the measured voltage from a modified speaker with a large hanging mass that is part of the sensor I use in a seismometer project. This is the transient response of the system when perturbed, showing the damped oscillations.

The setup for this measurement is shown in **Figure 2**. I used an analog front end to convert the induced current from the speaker into a voltage in the range an Arduino can measure with its analog pin.

From the measured data, I can fit the resonant frequency and q-factor for an ideal damped oscillator. The agreement of this simple ideal model and this real physical system is really remarkable.

At the heart of this process is bringing the data into the computer. While there are a few really cool low cost data acquisition systems like the DATAQ DI-145 Electronic Strip Chart Recorder ($29) and the more advanced LabJack ($108), I've been exploring using an Arduino as a data acquisition interface to the real world.

Even a low end RedBoard (UNO compatible from SparkFun for $19.95) has six independent analog-to-digital converters (ADC), each with 10-bit resolution. This makes Arduinos potentially great platforms for sensor data acquisition. When I started down this path to use an Arduino as a data acquisition system, the stumbling block for me was how to get the data from the Arduino directly into an analysis tool like Microsoft Excel.

I wanted a method that was easy, robust, low cost, and wasn't a long time-consuming process involving hacking a lot of code. Did I mention I wanted it to be easy?

The problem was not that I couldn't find any way of doing this. The problem was that I found *too* many ways of doing this. When I Googled "Arduino data acquisition," I got more than 250,000 entries. They generally fell into

Post comments on this article and find any associated files and/or downloads at
www.nutsvolts.com/index.php?/magazine/article/june2015_Bogatin.

three categories: in real time through the serial port; data logging into an SD card; or by Wi-Fi into the cloud.

While data logging or sending the data to a cloud server are really cool, for my first application I wanted to use my Arduino as a tethered data acquisition unit and suck out the data over the USB cable. Programming the Arduino to print data to the serial port — while there are a few timing limitations — is easy. It's getting the data from the serial port into an immediately useful format that was the challenge.

There were only 129,000 entries in Google under "Arduino serial data acquisition." The options for reading the data on the serial port ranged from simple programs written in Processing, Python, or C, to using high-end tools like LabVIEW and MatLab. (Did I mention I wanted an easy process?)

Then, there were the stand-alone tools that folks had written to read the serial terminal, parse the numbers, and display them in various ways. Some were free; some cost from $19 to over $99.

I spent time playing around with many of these tools. My criterion was I wanted to look at the data in real time as it came out of the Arduino, and display it in a high quality plot — preferably Excel compatible. Plus, I wanted the learning curve to be at most five minutes with minimal additional code I had to add to my Arduino sketches. (Did I mention I wanted it easy?)

After trying out more than 20 different tools, I settled on the best tool I found that met my criterion of being really easy and — best of all — it was free: PLX-DAQ.

The Simplest Way to Get Data from an Arduino into Microsoft Excel

PLX-DAQ is really a macro that runs in Microsoft Excel. It is free and can be

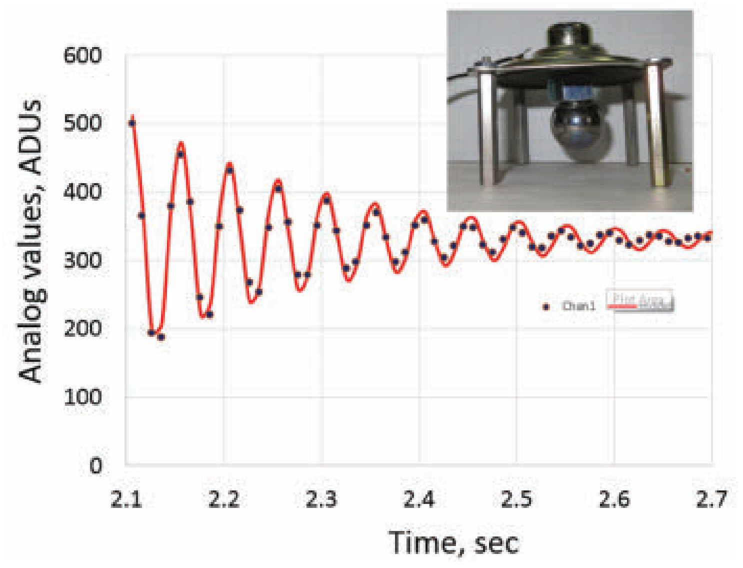


FIGURE 1. Measured response from the speaker and the fit to an ideal damped oscillator model. The agreement is excellent and even shows the onset of anharmonic behavior at small amplitude.

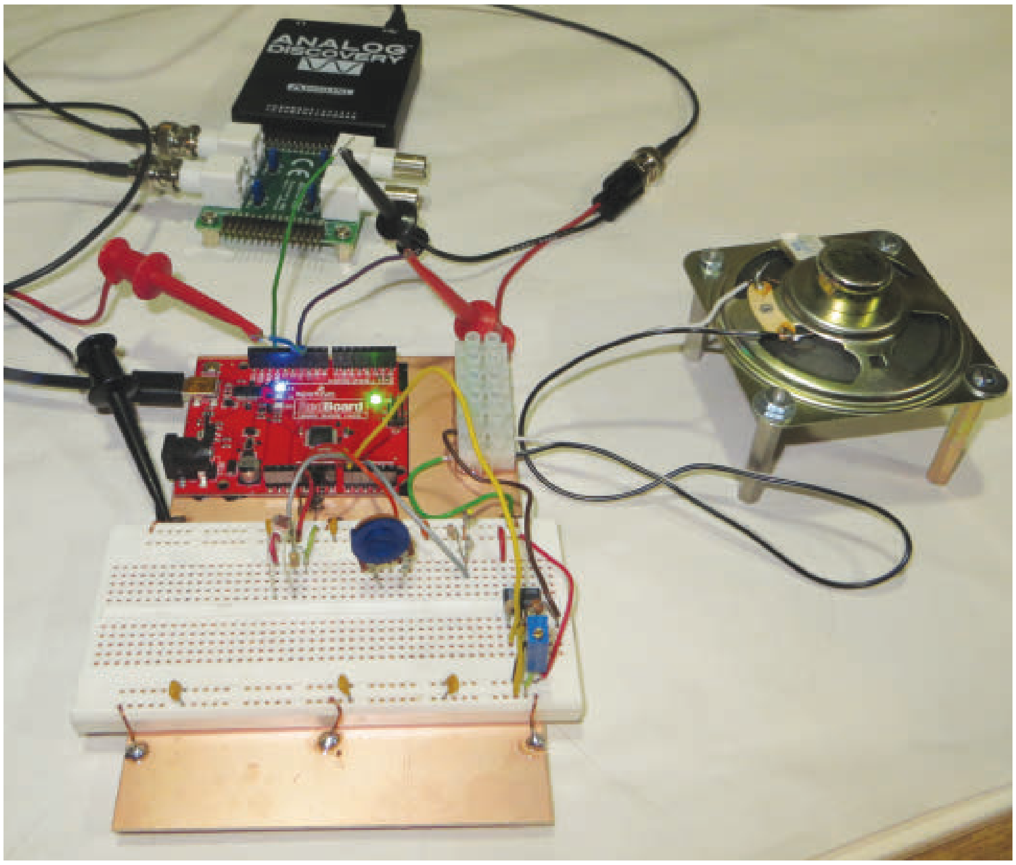


FIGURE 2. A complete measurement system consisting of the sensor (modified speaker), the analog front end (an op-amp mounted in a breadboard), the data acquisition system (my Arduino), and a scope used for diagnostics and debugging.

downloaded from Parallax, Inc. Ironically, it was originally written for Stamp microcontrollers — a competitor to Arduinos. Since it just reads the data coming in on the serial port, it's really independent of the specific microcontroller, as long as the correct command words are sent over the USB bus.

The Arduino writes successive rows of data separated by commas to the serial port using the *Serial.print ()* command. The word "DATA" has to be written in front of each row of data, with each column of data separated by commas. That's it. That's all that needs to be added to normal Arduino code to start plotting data in Excel using PLX-DAQ. **Figure 3** shows a code snippet to write data to PLX-DAQ.

I added a conditional flag to print the first two columns of data containing the number of averages for each data point and the specific time the data item was taken, only when using PLX-DAQ.

As a quick test, I looked at the data printed to the serial monitor as the three channels were read and printed to the serial port. **Figure 4** shows the serial monitor response — just as we expected. It has the right format to be read by PLX-DAQ.

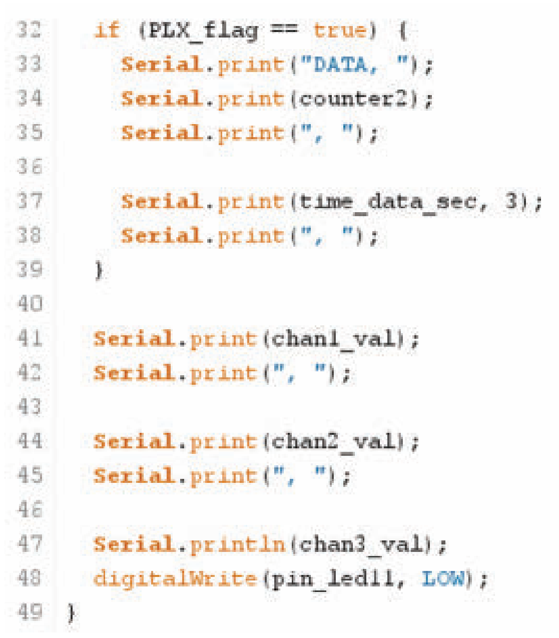
When you download and install the PLX-DAQ tool, it creates an Excel spreadsheet with an embedded macro. You have to enable the macro and allow it to run. The spreadsheet it opens up has three tabs, or worksheets. By default, PLX-DAQ will write all new data into the first worksheet. I set mine up to plot the values in the three data columns against the time column, using an X-Y scatter graph. This is just one of the multiple types of displays PLX-DAQ can create.

In the small dialog window that opens up in PLX-DAQ (shown in **Figure 5**), there are two settings you have to set: the COM port for the Arduino; and the baud rate. I always use the highest baud rate I can get away with. While most published example code I see uses 9600 baud, I never do this.

The highest baud rate I routinely get transferring into PLX-DAQ is 56000. This is my default value. In the *void setup()* function, I place the line *Serial.begin (56000);*. You also have to set this value in the PLX-DAQ dialog box.

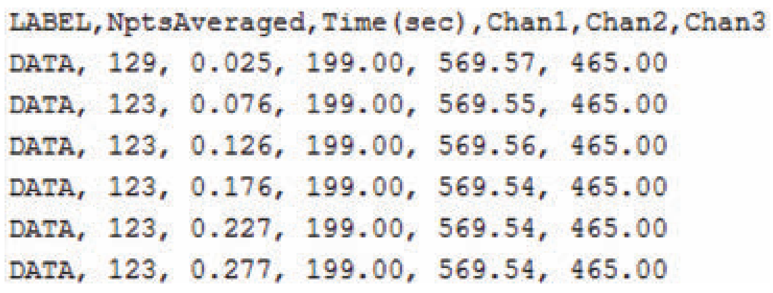
When you click the "connect" button, PLX-DAQ sends a *rest* command to the Arduino which starts the sketch from the beginning and catches all the transmitted data. This is effectively the start button.

Figure 6 shows an example of the Excel spreadsheet with the five columns of data and the plot of the three



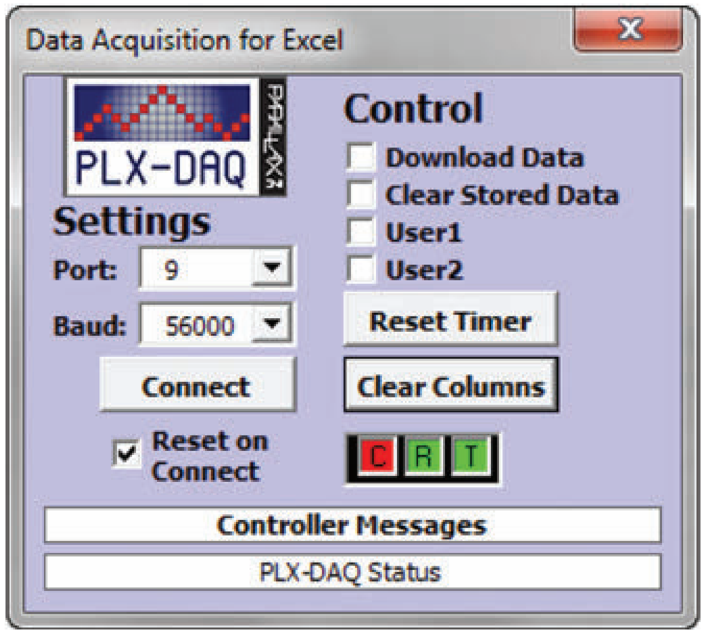
```
  if (PLX_flag == true) {
    Serial.print("DATA, ");
    Serial.print(counter2);
    Serial.print(", ");

    Serial.print(time_data_sec, 3);
    Serial.print(", ");
  }

  Serial.print(chan1_val);
  Serial.print(", ");

  Serial.print(chan2_val);
  Serial.print(", ");

  Serial.println(chan3_val);
  digitalWrite(pin_led11, LOW);
}
```

FIGURE 3. An example of the serial.print commands needed for PLX_DAQ to parse the characters on the serial port and place them in columns in a spreadsheet.



```
LABEL,NptsAveraged,Time(sec),Chan1,Chan2,Chan3
DATA, 129, 0.025, 199.00, 569.57, 465.00
DATA, 123, 0.076, 199.00, 569.55, 465.00
DATA, 123, 0.126, 199.00, 569.56, 465.00
DATA, 123, 0.176, 199.00, 569.54, 465.00
DATA, 123, 0.227, 199.00, 569.54, 465.00
DATA, 123, 0.277, 199.00, 569.54, 465.00
```

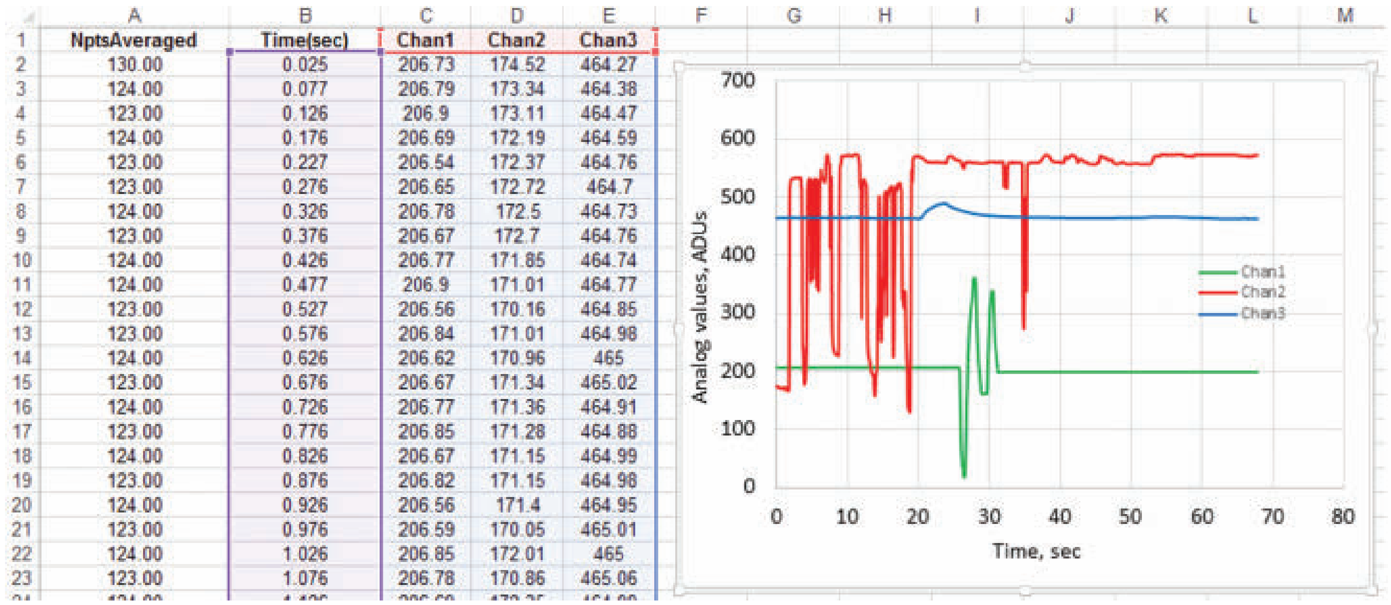
FIGURE 4. The characters printed to the serial port in the format read for PLX-DAQ to parse it into Microsoft Excel.



Data Acquisition for Excel

Settings
Port: 9
Baud: 56000
✓ Reset on Connect

FIGURE 5. The dialog box to set up PLX-DAQ. It automatically opens when you open Excel and enable macros. You only need to change the COM port and baud rate.



	A	B	C	D	E
1	NptsAveraged	Time(sec)	Chan1	Chan2	Chan3
2	130.00	0.025	206.73	174.52	464.27
3	124.00	0.077	206.79	173.34	464.38
4	123.00	0.126	206.9	173.11	464.47
5	124.00	0.176	206.69	172.19	464.59
6	123.00	0.227	206.54	172.37	464.76
7	123.00	0.276	206.65	172.72	464.7
8	124.00	0.326	206.78	172.5	464.73
9	123.00	0.376	206.67	172.7	464.76
10	124.00	0.426	206.77	171.85	464.74
11	124.00	0.477	206.9	171.01	464.77
12	123.00	0.527	206.56	170.16	464.85
13	123.00	0.576	206.84	171.01	464.98
14	124.00	0.626	206.62	170.96	465
15	123.00	0.676	206.67	171.34	465.02
16	124.00	0.726	206.77	171.36	464.91
17	123.00	0.776	206.85	171.28	464.88
18	124.00	0.826	206.67	171.15	464.99
19	123.00	0.876	206.82	171.15	464.98
20	124.00	0.926	206.56	171.4	464.95
21	123.00	0.976	206.59	170.05	465.01
22	124.00	1.026	206.85	172.01	465
23	123.00	1.076	206.78	170.86	465.06

FIGURE 6. The final result: The spreadsheet showing the data as it came in and the resulting plot of the three channels of ADCvalues (0 to 1023) using the time stamp data. This plot can be modified using standard Excel tools.

analog channels. I created the plot using the built-in Excel functions so that any data written into the columns D, E, and F would be plotted in real time as it came in, using column B — the time — as the X axis.

A Little Finesse — Making Life Easier

It really took me less than five minutes after opening PLX-DAQ for the first time before I had data in Excel and plotted. Now, I was ready to add a few simple features to clean up the data.

PLX-DAQ understands a few code words. The word DATA in front of each line of data (separated by commas) is the command to place each data value in a separate cell on the same row.

The line feed *Serial.println()*, as the last line in the *serial.print* statements, also prints the line feed and tells PLX-DAQ to add the next set of data to a new row.

There are two other commands I find very useful. The first is *CLEARDATA*. When this key word is printed to the serial port, PLX-DAQ will clear the cells in the Excel spreadsheet and move its pointer to start adding data to row 1 in the first worksheet. This means I can use the same worksheet over and over again — especially if I have a graph already formatted the way I want.

When I have a set of data collected and I want to keep it, I make a copy of it and place the copy in the first sheet position. This is done with a right mouse click on the worksheet tab and selecting the *copy* command.

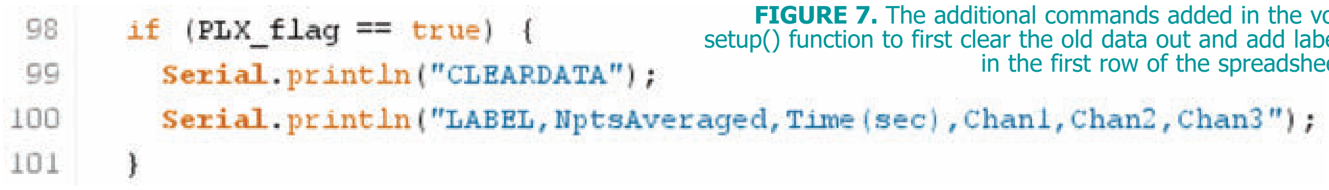
Now, I have a new worksheet ready to collect and plot data with formatted plots, but it's full of old data. When the *CLEARDATA* command is sent, the old data in this new spreadsheet is cleared out and I start over with a blank worksheet. Fortunately, a plot is already formatted, waiting for new data to plot.

The second command that makes my life easier is *LABEL*. I like to have my data as well documented as possible. In this simple example, I wanted to display five columns of data, the number of points averaged per data value, the real time each data point was taken, and the three analog values from each channel. I used the *LABEL* command to send the labels for each column (separated by commas) as the first command.

This command only needs to be sent once, to set up the column labels. I put this in the *void setup()* function, so it executes right at the beginning of the sketch. Since I wanted a little flexibility on how I use the serial port, I added the flag to only print the *LABEL* command and values if the *PLX_flag* is set for true. These commands are

Avoid this common problem: While the standard baud rate for the serial port is 57600, PLX-DAQ only offers the option for 56000. If you use 57600, some of the data read by PLX-DAQ will be garbled. You must set up the baud rate in the Arduino sketch with the same value as you select in PLX-DAQ. I recommend the highest value of 56000 as your default.

Avoid this common problem: Only one device can use the serial port at a time. The most common mistake I make is having the serial monitor open to read the serial port while I want to have PLX-DAQ read the data. I get an error in this case. Wait for the code to be uploaded to the Arduino and be running. Make sure nothing else is talking over the same COM port, and then click the "connect" button. Likewise, to talk to the Arduino, be sure to click the "disconnect" button on the PLX-DAQ dialog box to free up the serial port connection for the Arduino.



```
if (PLX_flag == true) {
  Serial.println("CLEARDATA");
  Serial.println("LABEL,NptsAveraged,Time(sec),Chan1,Chan2,Chan3");
}
```

FIGURE 7. The additional commands added in the void setup() function to first clear the old data out and add labels in the first row of the spreadsheet.

Avoid this common problem: Since the *CLEARDATA* command erases all the data in the spreadsheet, make sure you either want to remove the data or have a new worksheet set up in the first tab position before you click the "connect" button.

shown in **Figure 7**.

With these two additional features, I can set up a worksheet exactly the way I want it with the graph formatted using all the built-in cool features of Excel, and I can use this worksheet over and over for each new run of measurements. The old data in the copied sheet is cleared out, the column headings are added, and all the data flows into the sheet in real time.

Measuring the Timing of Arduino Operations: a New Window for Debugging

Now that I could effortlessly bring data into a spreadsheet, I focused on the quality of the data and how fast I could take it.

A key feature of a data acquisition system is to have an accurate time associated with each data point. I didn't just want a lot of data points; I wanted a lot of data at well-defined instants of time.

The best way of capturing the acquisition time is to take advantage of the built-in timer functions, *millis()* and *micros()*. They return the current elapsed time since the Arduino was turned on in milliseconds and microseconds, respectively.

As long as I record the system time just before and just after an analog measurement, I can get an accurate measure of the actual time stamp for each data point. The challenge was to determine how fast I could collect and write the data.

Most online forums suggest using the built-in timer functions like *millis()* or *micros()* to calculate the execution times and print the results to the serial port. You just have to add debug code to measure the time intervals between sections of your code, print the times out to the serial port, and read them with the serial monitor.

While this is a useful technique, I am a measurement guy and like to use my scope and digital *write* pins to get useful timing information of Arduino code. I think this is an important technique as it gives a very visual window into the time it takes for sections of code to execute. I use

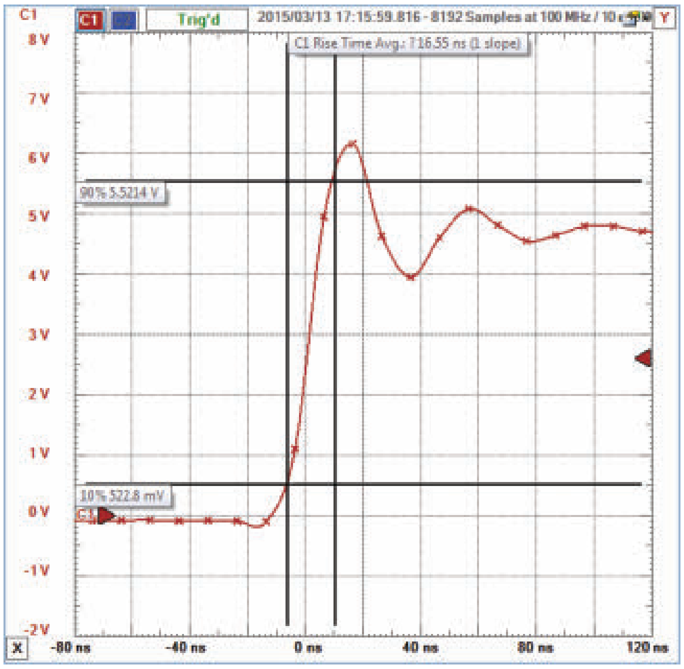


FIGURE 8. Screenshot from the Analog Discovery scope showing the measured 10-90 rise time of the Arduino digital signal of 16 nsec. This is limited by the scope. Its actual rise time, measured with a faster scope, is 3 nsec.

Avoid this common problem: Even if you initialize a timer as an unsigned long, the largest integer value it can hold is 4.3 x 10^9. If this is in microseconds, the longest time you can record before the timer resets is 4.3 x 10^3 seconds, which is about 71 minutes. If you want to record data with a time stamp for a longer time than 71 minutes, use the *millis()* timer. This can count up 1000x longer, or 49 days before resetting.

Links

www.dataq.com/products/di-145/

http://labjack.com/u3

www.sparkfun.com/products/12757

www.parallax.com/downloads/plx-daq

www.digilentinc.com/Products/Detail.cfm?NavPath=2,842,1018&Prod=ANALOG-DISCOVERY&CFID=7824927&CFTOKEN=628690e39396761f-CE94CA68-5056-0201-02093BB386E8F629

www.sparkfun.com/products/250

www.sparkfun.com/products/9088

it to debug my code all the time.

My favorite scope is the Digilent Analog Discovery scope. This is a two-channel, 10 MHz bandwidth, 100 Msample/sec, USB scope with 14-bit vertical resolution and a built-in two-channel arbitrary wave generator. It also has a 16-channel logic analyzer. The user interface is incredibly easy to use and incredibly versatile, and at $279, it's a steal.

The 10 MHz analog bandwidth is a perfect range for the Arduino. With this receiver bandwidth, we can see rise times on the order of 0.35/10 MHz = 35 nanoseconds. **Figure 8** is a screenshot of the zoomed in rising edge of a toggling digital output pin on my Arduino showing a measured 10-90 rise time of about 16 ns — well below the estimated limit of 35 ns. Note that the acquisition rate is only 100 Msamples per second (or one sample every 10 ns), so there is a bit of interpolation used to get the 16 ns rise time. This is plenty fast for most analog applications.

The RedBoard Arduino has an onboard 16 MHz clock generated by a very precise crystal. A single clock cycle is about 60 ns. In another experiment, I found that by using the *PORTB* command, I could toggle eight of the digital output pins off and on in four clock cycles, or in 250 ns. This sort of time is overkill for any typical data acquisition I had in mind, and I wanted to leverage the *digitalWrite* functions.

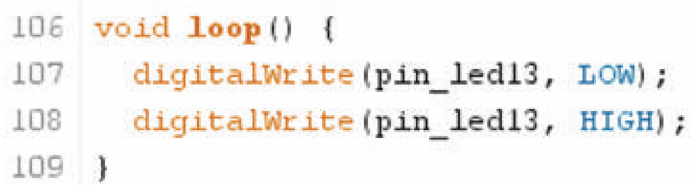
The first question I wanted to explore was just how fast can I toggle an output pin off and on using the *digitalWrite* command. I set up a sketch to just have the two commands *digitalWrite(13, LOW)* and then *digitalWrite(13,HIGH)* inside the *void loop()* function to turn output pin 13 off and then on. This code is shown in **Figure 9**.

I used the venerable pin 13 which also flashes the LED on the board. **Figure 10** is a snapshot of the screen of the Analog Discovery scope showing the voltage at the output of pin 13. The pin is set low, then the next command is to write it high. In the scope trace, you can see the low lasts for 5.25 µs. This is about 90 clock cycles. Did I mention that the Analog Discovery scope also has over 50 math functions allowing high resolution waveform feature measurements? In this case, I extracted the off time and on time of the digital signal using a simple math function and displayed the extracted values in real time.

In this measured output, we see that the pin stays low for 5.25 µs, and then the output *HIGH* command is executed. This lasts for 7.18 µs, which includes the execution time for the *loop* command. Overall, this loop is executed in 12.43 µs. Altogether, this is about 200 cycles. This is plenty fast for my testing.

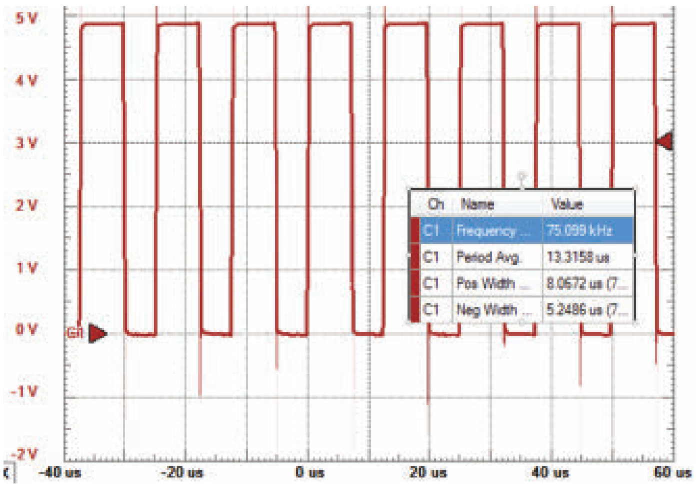
Adding comment lines to the code has no impact on the execution time. In fact, I wrote the *toggle* commands in a function and just called the function in the *void loop()* function. The execute time went up to 12.62 µs. This is only 190 ns of overhead to execute a function call, or about three clock cycles. Function calls are pretty fast.

We can use the overhead time as a starting place estimate; the overhead time to turn off pin 13 and turn it on, that's about 12.4 µs. Now, we can start using it to help decode the timing for executing other functions like reading the analog pins and writing to the serial port.



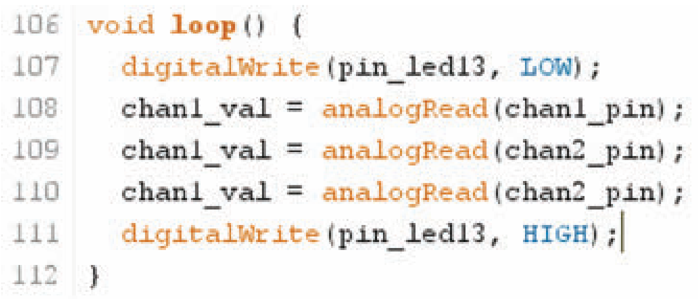
```
void loop() {
  digitalWrite(pin_led13, LOW);
  digitalWrite(pin_led13, HIGH);
}
```

FIGURE 9. Code snippet to toggle pin 13 off then on as fast as possible.



Ch	Name	Value
C1	Frequency ...	75.099 kHz
C1	Period Avg.	13.3158 us
C1	Pos Width ...	8.0672 us (7...
C1	Neg Width ...	5.2486 us (7...

FIGURE 10. Screenshot for the Analog Discovery scope measuring the voltage on pin 13, showing the on and off time. Inset is the on-screen real time measurement of the waveform features.



```
void loop() {
  digitalWrite(pin_led13, LOW);
  chan1_val = analogRead(chan1_pin);
  chan1_val = analogRead(chan2_pin);
  chan1_val = analogRead(chan2_pin);
  digitalWrite(pin_led13, HIGH);
}
```

FIGURE 11. Code snippet to use the digital write pins as timers when reading three analog channels.

How Long Does It Take to Read an Analog Pin?

I set up my initial experiments to read three analog channels. I connected some simple sensor inputs just so I

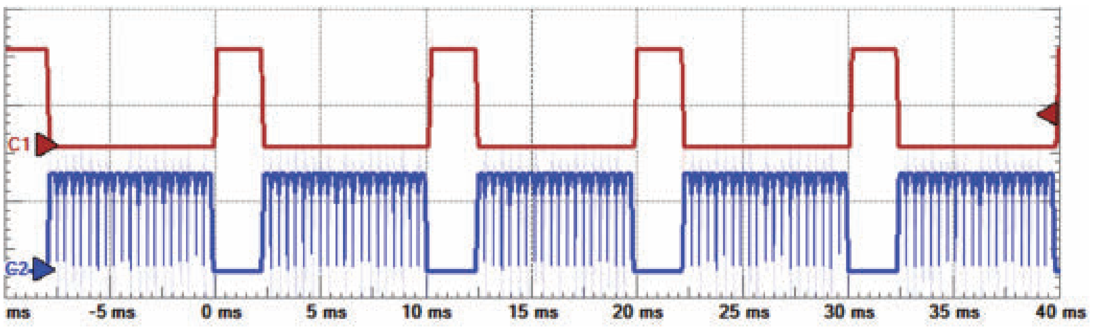


FIGURE 12. Measured timing diagram showing the relative time to print to the serial port (top trace, on time) and the successive analog channel reads (bottom trace, on times). This shows the many consecutive analog reads which were then averaged to arrive at one value per channel in each sample interval.

had something fun to offer as signals. In channel 1, I had a simple one-turn pot connected between the +5 and gnd.

In channel 2, I had a thermistor connected to +5 with a 10K resistor. All resistors change their resistance with temperature. How much the resistance changes is described by the temperature coefficient of resistance (TCR). For a carbon resistor, it's about 0.1% per degree C. A thermistor turns this problem into a feature. Its TCR can be as high as 5% per degree C.

The thermistor I used from SparkFun was nominally 10K ohms with a 5% change per degree C. Touching it increased its resistance by 20-40%. I turned it into a simple resistor voltage divider between +5 and gnd.

In channel 3, I added a simple light sensitive resistor, also from SparkFun. Its nominal resistance in the dark was about 10K ohms, and it could drop to 2K ohms in the light. Again, I just added it to a simple voltage divider and measured the voltage across the 10K resistor in series.

Using my *digitalWrite* triggers as a timer, I measured the time it took to read a single analog channel, and then to read all three channels sequentially. The code to do this is shown in **Figure 11**. Reading one analog channel increased the time for the low signal to be on from 5.25 µs to 120.7 µs. This is about 115.5 µs for an analog *read* command. When reading three channels consecutively, the low bit time increased to 344.3 µs. Taking out the *digitalWrite* overhead, this is (344.3 – 5.25)/3 = 113 µs per channel — very close to the measurement of the single analog read. To be generous, I use the value of 120 µs as the time to read an analog channel.

In principle, if there are no other functions going on, it sounds like I could take one channel's worth of data at a sample rate of about 8 kHz. However, it's what we do with the data that will really slow things down.

How Long Does it Take to Perform a *Serial.print()*?

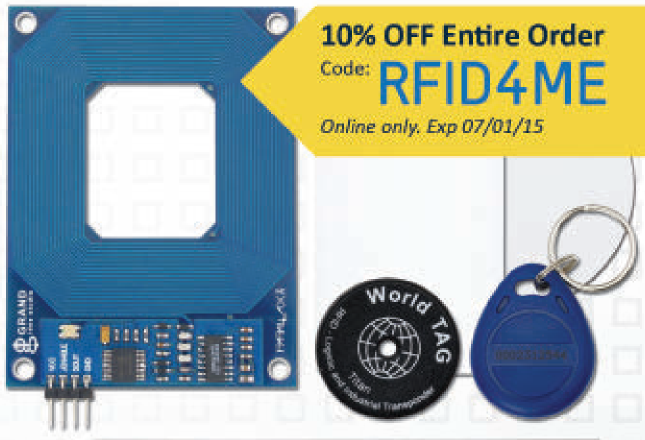
For this first data acquisition system, I wanted the fastest data acquisition I could easily get and have the data go into an Excel spreadsheet with PLX-DAQ. The limitation I quickly found was the *print* command to the serial bus.

The first way to decrease the time to print to the serial bus is to use the highest baud rate practical. The difference in times to print using 9600 and 56000 is almost (but not quite) the x5.8 difference in baud rate. There is some overhead used in the *serial.print* command.

Printing one character at 56000 takes about 0.4 ms. However, printing 30 characters takes about 5.6 ms. Depending on the number of characters in the data being printed, the total print time in my code may vary from 2 to

RFID Readers

For access control, automatic identification, robotics, navigation, inventory tracking, etc.



10% OFF Entire Order
Code: RFID4ME
Online only. Exp 07/01/15

PARALLAX INC
www.parallax.com

In accordance with U.S. Patent Nos. RE40,145; 5,817,207; 6,036,099; 6,214,155; 6,514,367; and 6,557,766.

7 ms. This means that a safe sample interval is 8 ms.

Getting the Most Out of the Measurements by Averaging

Based on the variation in the time for the *print* command, I decided to use a short sample interval of 10 ms. This was reasonable for my typical application and also robust, leaving plenty of time (even in the worst case) for the *serial.print* command to execute.

If the *print* command took only a few ms, this left a lot of dead time. I decided to take advantage of this by using the available time in the sample interval to repetitively read the analog channels and average the results. What is exported to the *serial.print* command at the end of each time interval is the average of all the readings during the interval.

When the time available to sample was on the order of 7.5 ms and it only took about 0.33 ms to read the three channels, it was possible to perform as many as 7.5/0.33 = 20 readings for each exported data point. This helps reduce the noise.

I used output pin 12 to toggle on when I started to read the first analog channel and toggle off when I finished reading the last channel. I used pin 11 to turn on when I started the *serial.print* operations and toggle off when I finished. **Figure 12** shows the measured timing diagram when the sample interval is 10 ms.

Conclusion

I like data and I like easy. I started this project with a goal to find a simple, robust, and easy way of bringing data from my Arduino into Microsoft Excel. After a lot of evaluation, I decided on PLX-DAQ as the tool to read data on the serial port directly into Excel. This turned out to be a perfect solution.

Along the way, I found that a valuable debug tool was to use a few digital *write* pins as a new window into the timing associated with different functions. This gives a quick graphical view of when the code executes operations and how long it takes.

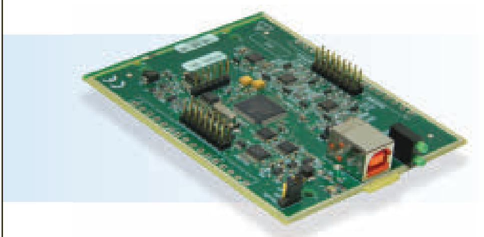
This solution turns the very low cost Arduino into a very robust, versatile, and accurate data acquisition system capable of as fast as 100 points/sec, plotted directly into Excel in real time. It's a great starting place to feed my need for data. **NV**

M E A S U R E M E N T   C O M P U T I N G

OEM & Embedded DAQ

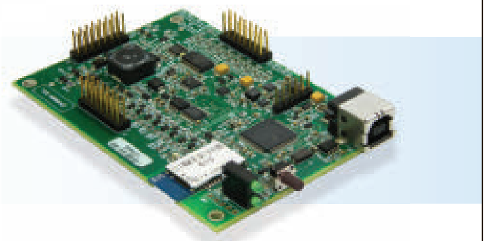
from concept to product

Easy to Use • Easy to Integrate • Easy to Support

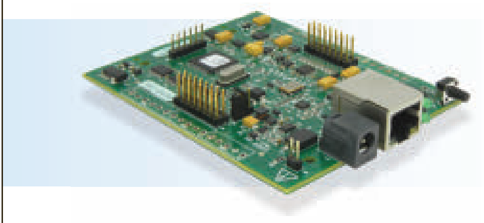


USB Multifunction
USB-201-OEM
***Only* $99**
- 12-bit, 8 SE analog inputs
- 100 kS/s sampling
- 8 digital I/O, 1 counter
- Support for Windows®, Linux®, and Android™

Wireless Multifunction
BTH-1208LS-OEM
***Only* $189**
- Eight 11-bit SE, or four 12-bit DIFF analog inputs
- 1 kS/s sampling over Bluetooth, 50 kS/s sampling over USB
- 2 analog outputs
- 8 digital I/O, 1 counter
- Support for Windows®, and Android™



Ethernet Multifunction
E-1608-OEM
***Only* $479**
- 16-bit, 8 SE/4 DIFF analog inputs
- Up to 250 kS/s sampling
- 2 analog outputs
- 8 digital I/O, 1 counter
- Support for Windows®, and Android™



Contact MCC for a no-risk 30-day evaluation and test drive.

mccdaq.com/OEM

MEASUREMENT COMPUTING™

Contact us
1.800.234.4232

Using Serial Bluetooth

By Craig A. Lindley

In a previous article ("Driving LEDs with Microcontrollers," *N&V's* April 2015 issue), I developed microcontroller-based hardware and software to control a matrix of 8x8 RGB LEDs. I thought since I still had that hardware laying around, I would use it as a test bed for experimenting with Bluetooth. I had never done anything with Bluetooth before, so I thought it was about time I did.

As you may know, Bluetooth is a standard for the short-range wireless interconnection used in cell phones, computers, and other electronic devices. According to Wikipedia, "It was originally conceived as a wireless alternative to RS-232 data cables." Of course, today, Bluetooth is used for many purposes, including:

- Hands-free control and use of cell phones.
- Streaming of music for the home, in cars, and even for wireless headphones.
- Streaming of data for file transfers between phones and PCs, and PC to PC.
- For wireless keyboards, mice, and printers.
- For wireless tethering, where "tethering" is the act of sharing a device's network connection with another device. Most tethering is done via Wi-Fi, but it can also be done with Bluetooth. An advantage of Bluetooth tethering is that it requires much less power than the equivalent Wi-Fi connection.
- And yes, streaming of serial data as an alternative to RS-232 cables.

It is this last purpose that we will be experimenting with in this article.

Abbreviated Bluetooth History

In the early 1990s, a group of engineers at the Swedish company, Ericsson developed the technology that would later be called Bluetooth. The name came from a Danish king who in English is called Harold Bluetooth. Harold united warring factions in his kingdom, just as Bluetooth technology united technology companies in their pursuit of a short-range data communication standard. No one company owns the Bluetooth technology. A Special Interest Group (SIG) of technology companies work together to maintain, extend, and promote Bluetooth. As mentioned, the original intent of Bluetooth was for the replacement of RS-232 cables, but we all know Bluetooth is much more than this.

The original Bluetooth technology is today referred to as classic Bluetooth because of the arrival in 2011 of Bluetooth LE, or Bluetooth Low Energy (BLE). BLE shares many of the same attributes of classic Bluetooth, but is targeted for the type of applications that require extreme low energy usage. In fact, energy so low that a coin battery like a CR-2032 could power a BLE device for five to 10 years.

Classic Bluetooth was designed to continuously stream data, whereas BLE was designed for periodic or episodic data transfer as would be required for remotely located sensors, for example.

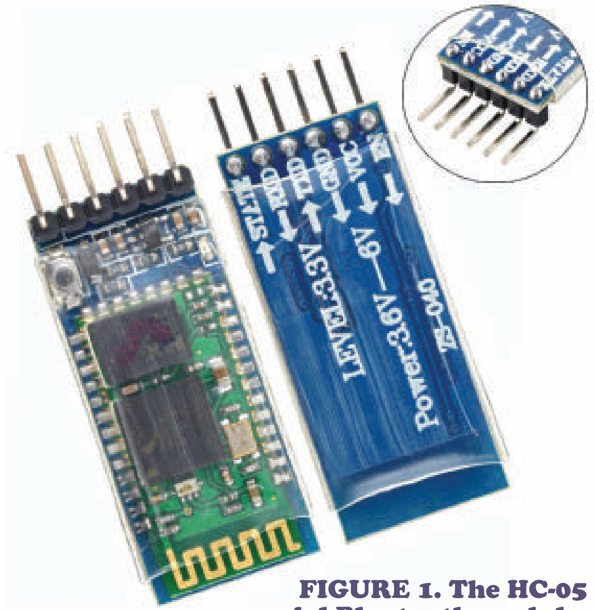


FIGURE 1. The HC-05 serial Bluetooth module.

with a Microcontroller

A downside of BLE is that it only supports data rates up to around 100 Kbps whereas classic Bluetooth devices can communicate up to 2 Mbps. The new Bluetooth 4.2 revision blurs the lines between classic and low energy Bluetooth even further.

Bluetooth Usage

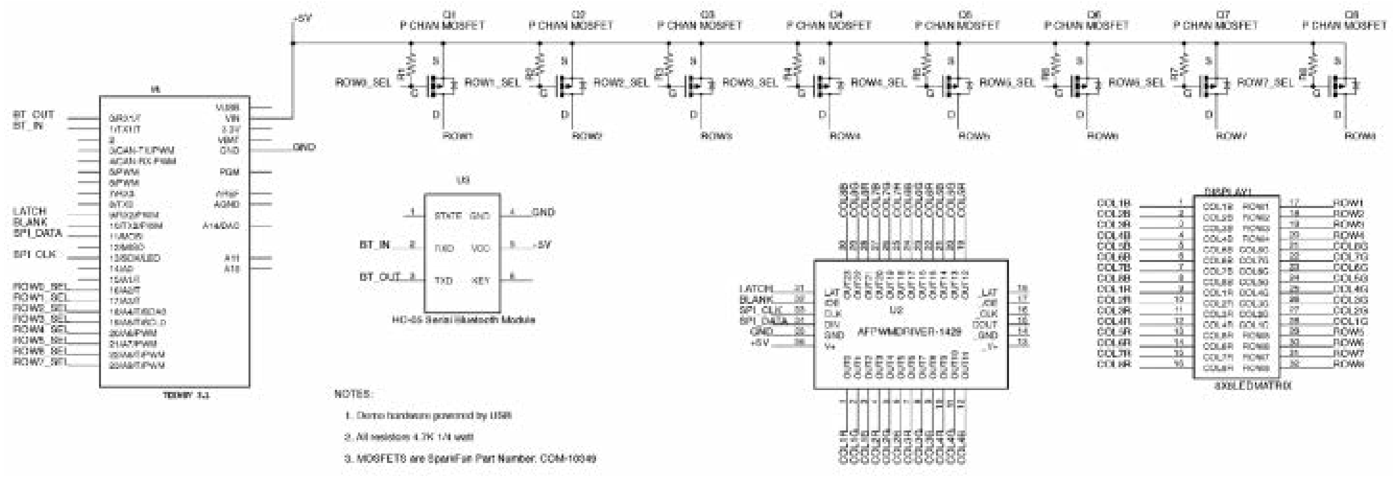
Bluetooth (BT) devices must go through the process of "pairing" before they can be used together. Not all BT devices will pair together, however. For example, it wouldn't make sense to connect a BT mouse to a BT headset, although you should be able to pair a BT headset to a BT phone, or a BT mouse to a BT enabled computer.

Each BT device has one or more use "profiles" which describe the types of services they can provide. Devices that are meant to pair together share the same profile. A current iPhone, for example, has seven different BT profiles but curiously lacks the SPP (or Serial Port Profile) we require for our use here. The SPP defines how to set up virtual serial ports and connect two BT enabled devices. Since it has been determined that two BT devices are compatible for connection, a passkey will be required to complete the pairing. Passkeys are generally simple numeric strings like 0000 or 1234. The Bluetooth module used for our experiments uses 1234 for its passkey, though this can be changed for security reasons. Yes, I was quite surprised when I found out that I couldn't use my iPad for these experiments because it doesn't support the SPP profile. Luckily, my Nexus 7 Android tablet does.

Bluetooth Connected Hardware

As I mentioned, I wanted to use the 8x8 RGB LED matrix hardware I developed previously for my Bluetooth experiments. As you may recall, this hardware used a Teensy 3.1 microcontroller (from **pjrc.com**) to control an LED matrix using multiplexing. My thought was I would get an inexpensive Bluetooth module and connect it to the Teensy, then use my Nexus 7 tablet to control things. This actually worked out to be much easier than expected.

Since I was going to use Bluetooth as "a replacement for an RS-232 cable," I had to choose a Bluetooth module that could do so. Looking around on the Internet, I came across the HC-05 serial Bluetooth module (**Figure 1**). I subsequently got one for $8 (with free shipping) on eBay. The HC-05 module is capable of being a Bluetooth master or a Bluetooth client; we will only be using it as a client. Only four connections are required to use the HC-05 module in its simplest configuration, with power and ground being two of them. The two remaining signals — RXD (receive data) and TXD (transmit data) — are connected to the Teensy 3.1 microcontroller as shown in the schematic in **Figure 2**. The TXD signal from the HC-05 is connected to the Teensy's RX1 signal (pin 0), and the HC-05's RXD signal is connected to the Teensy's TX1 (pin 1). The other HC-05 pins are not required for our use here.



NOTES:
1. Demo hardware powered by USB
2. All resistors 4.7K 1/4 watt
3. MOSFETS are SparkFun Part Number: COM-10349

FIGURE 2. Schematic shows connection between a HC-05 serial Bluetooth module and the microcontroller hardware developed in a previous article.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/june2015_Lindley.**

Mood Light mood@	Random Patterns rand@	Sequential Patterns seq@
Off off@	Speed speed@	Off off@
Hue hue@	Saturation sat@	Value val@
Message 1 msg This is a test message @	Message 2 msg Hello Nuts and Volts @	Message 3 msg Hello World @

Table 1.

A red LED will begin blinking as soon as power is applied to the HC-05 module. This indicates the module is operational, but not yet paired with another Bluetooth device. NOTE: We will be using the HC-05 with its default configuration. If you are interested in changing its configuration, please consult the following: **www.instructables.com/id/Modify-The-HC-05-Bluetooth-Module-Defaults-Using-A/?ALLSTEPS**.

Software

There are two parts to the software that need to be discussed: the software on the device we will be using to control the LED matrix; and the software which will run in the Teensy 3.1 that receives commands from the controlling device and executes them.

If you go to the Google Play app store, you can find many apps that support the Bluetooth SPP profile. There are too many to list here of both the free and paid varieties. The app I decided to install on my Nexus 7 is a freebie called "Bluetooth SPP Tools Pro" (**https://play.google.com/store/apps/details?id=mobi.dzs.android.BLE_SPP_PRO&hl=en**) which was written by a fellow named Jerry Li. This app is more than capable of controlling the LED matrix remotely.

When first started, this app will scan for any Bluetooth devices in the area and (with our hardware connected and running) it will find the HC-05 device. The first time the HC-05 is detected, you will have to enter the "1234" passkey to complete pairing. If you look at the flashing red LED on the HC-05 module, you will see that the cadence of the flashing changes when the device is paired.

On the screen presented after pairing, you select the mode you wish to use for communication with a remote BT device. I chose to use the Keyboard mode which provides an array of 12 buttons that are configured to send BT commands when clicked. I configured the buttons as shown in **Table 1** (the top line of text is the button label/name and the bottom line is the command string that will be sent when the button is clicked).

The buttons are configured by clicking the three vertical dots at the top right of the window and then selecting the *Button set* menu item. Then, when you click a button, you get a screen asking for the *BTN Name* (which is the top line above) and *BTN Down* into which you input the bottom line. When you are finished configuring the buttons, click the dots again, but this time select *Button set complete*. You should then click *Save2File* to save your configuration.

Okay, so what do these button commands do? When you click a named button, the string configured for the button is sent wirelessly to the HC-05 module which, in turn, sends it serially to the Teensy 3.1 microcontroller. Code in the Teensy parses the received message string and reacts accordingly. **Table 2** describes the effect of each button click.

The software running in the Teensy controller is rather complex and will need to be studied to be understood. The first thing to note is that the BT data from the HC-05 is received on the Teensy's serial 1 port. I used the *SerialEvent* library (see **https://github.com/duff2013/SerialEvent**) to handle this data because it runs completely in the background and only calls the *btDataReceived* function in the code when string data with the @ delimiter is received. This means the Teensy is fully available for running the patterns and will only be interrupted when BT data is received and in need of processing.

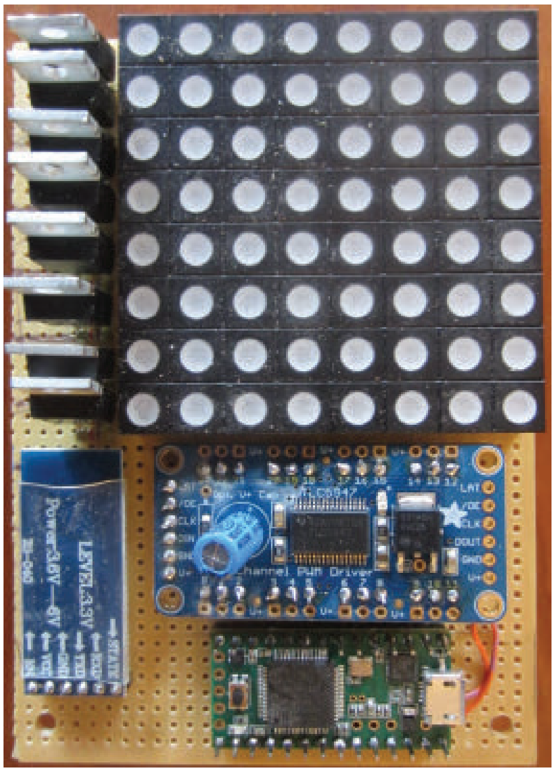


FIGURE 3. The LED matrix hardware with the addition of the HC-05 serial Bluetooth module.

The Teensy code is written as a series of state machines so that the display patterns can run at their full rate while constantly checking for the arrival of BT data. The *loop()* function in the sketch is looping many times a second, running the top level state machine which, in turn, calls other lower level state machines. This design approach allows the display patterns to be active and lively while still allowing the Bluetooth user interface to be very responsive. Again, study the code if you want to understand it or just use the code as-is and enjoy it. Your choice. To program the Teensy 3.1 controller for this project, you will need to have the Teensyduino environment installed on your development computer.

See **http://pjrc.com/teensy/td_download.html** for how this is done. You will also need to make sure your Arduino IDE (integrated development environment) is version 1.0.5 or newer.

The code for this project is in a zip file called Lindley-BluetoothMicrocontrollers.zip at the article link. This file should be unzipped and the *SerialEvent* directory and all of its contents should be moved to your Arduino libraries directory. The *BTLEDMatrix8x8* directory from the zip file should be moved into your Arduino project directory. Your Arduino IDE will need to be restarted to see the new library. Once that is done, open the Arduino IDE and using the File/Sketchbook menu selection to navigate to the *BTLEDMatrix8x8* sketch. Load it, compile it, and download it to your Teensy controller.

Once operational, pair the HC-05 module on the LED matrix with your BT device, and you will be ready to go.

The Crystal Palace

As I finished writing the software for this article, I was faced with a quandary. I had the LED matrix being successfully controlled via Bluetooth and I was really quite pleased with the result. I could run some very interesting and colorful patterns; I could use the device as a mood light; and I could even display scrolling text messages. What was I to do? Tear down the breadboard I had just finished building to scavenge the parts to use elsewhere, or should I try and figure out a use for the finished product. I, of course, chose the latter.

I started thinking about fiber optic displays and how light from a source would travel up a piece of fiber optics and would shine out the other end. I had some clear acrylic rod in the garage, so I took a short piece, sanded its ends to diffuse the light, and placed it vertically on the 8x8 RGB LED matrix. I was pleased to see the light provided by the RGB LED under the rod would shine up through it and brightly illuminate the end, while the rod itself would take on the color very subtly. This really got my juices flowing.

I could mount the LED matrix horizontally and use it as a light source for a group of acrylic rods positioned vertically over it. In other words, the rods would sit directly on the LED matrix.

Next, I needed to determine the arrangement of the rods over the LED matrix. My first thought was one rod for each LED. I could use one diameter of rod of all the same length, or I could vary the lengths to give some up and down motion to the light. I liked the idea of varying the rod length but the uniformity of all the same rod diameters didn't appeal to me.

In the end, I decided to use a large diameter rod in the center of the display (conveying and mixing the light from many LEDs), and use two different sizes of rods around the perimeter of the LED matrix. I decided on the

Mood Light	This mode makes the LED matrix function like a mood light. It displays a constant color; the hue, saturation, and value of which can be set with other buttons. If you feel *orange* in the early evening, you can set the appropriate color for the mood light.
Random Patterns	There are 12 colorful and lively lighting patterns coded into the Teensy software. By clicking this button, these patterns are selected randomly and run for a set period of time until the next pattern is randomly selected.
Sequential Patterns	Clicking this button causes the 12 patterns to be run sequentially.
Off	Clicking this button causes the LED matrix to go off. Clicking any of the top three buttons or the message buttons will cause it to come back on.
Speed	This button controls the speed of the lighting patterns in Sequential Pattern mode. Clicking this button repeatedly causes the speed to increase to maximum, and then return to minimum speed and begin increasing again. Speed is picked randomly in the Random Pattern mode.
Off	I didn't have any other functions I wanted to implement, so this button also turns the LED matrix off.
Hue	This button changes the hue while in the Mood Light mode. Clicking this button moves the mood light color across the visible spectrum and then starts over again.
Saturation	This button changes the saturation of the LED matrix's color while in Mood Light mode. Colors with high saturation are pure. As saturation is lowered, the color tends towards white. Like many of the other buttons, the saturation value increases to a maximum, and then resets to minimum and rises again.
Value	This button changes the value of the LED matrix's color while in Mood Light mode. Colors with a high value are bright, while colors with a low value tend towards black. Like many of the other buttons, value increases to a maximum, and then resets to minimum and rises again.
Message 1	Pressing this button causes whatever message configured for the button to be displayed on the LED matrix in a scrolling manner using a white font.
Message 2	Pressing this button causes whatever message configured for the button to be displayed on the LED matrix.
Message 3	Pressing this button causes whatever message configured for the button to be displayed on the LED matrix.

Table 2.

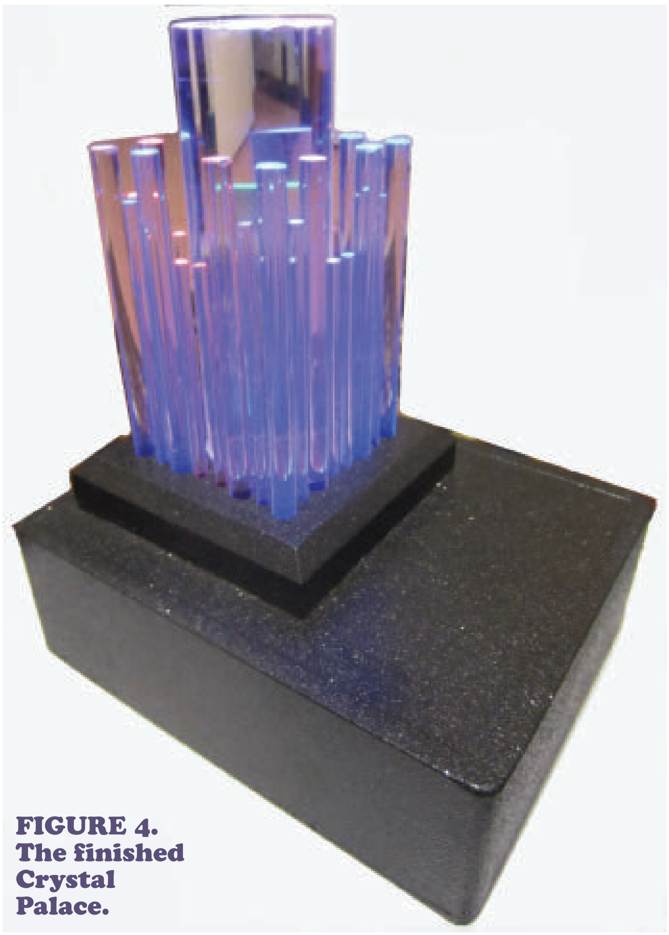


FIGURE 4. The finished Crystal Palace.

one LED to one rod relationship around the perimeter. With this arrangement, I would have light from the display patterns moving in both the X and Y directions across the LED matrix, and I would have light moving up and down in the Z direction as well, because of the varying lengths of acrylic rods. In addition, I would have three different sizes of rod ends, like little glowing discs that would be lit from the LEDs below.

When I finally visualized how this thing might look and how it could be built, I immediately flashed back to a Superman movie and his crystal palace hide-away. Thus, the Crystal Palace is the name I gave to this device. So, off I went to Delvie's Plastics (**delviesplastics.com**) and ordered 1/8", 1/4", and 1-1/2" acrylic rods. Delvie's has the best prices for plastic I have seen.

However, because you have to buy the rods in quantity, this can be a kind of expensive purchase. The upside it that I have enough material to build many projects that require acrylic rods.

Next up, I had to figure out how to position the acrylic rods accurately over the LED matrix and support them well enough that they stood vertically. Being a wood worker, I decided to mill a piece of 3/4" hardwood with holes for each rod. This worked out well.

Finally, I built an enclosure for my breadboard out of 1/4" MDF and glued the drilled hard wood support on top, after carefully measuring so that the rods would be positioned accurately over their respective LEDs in the matrix. I also cut a square hole in the side of this enclosure for the USB power cable which connects to the Teensy 3.1 microcontroller to pass through. After gluing up the enclosure and doing some finish sanding to round over edges and remove any traces of glue, I painted the enclosure with black sparkle paint. This turned out to be a crude but effective enclosure (**Figure 4**).

You can watch the finished Crystal Palace in operation at **www.youtube.com/embed/mtjrkF2MqbM**. If I were to build another one of these, I would do some things differently.

First, I would have made the breadboard square and not rectangular by mounting components on both sides of the board. I might even make a PCB (printed circuit board) instead of using point to point wiring next time.

Second, I would probably design the rod support and enclosure in a CAD system and 3D print it for much better positional accuracy.

Finally, I am thinking about making a bigger, better Crystal Palace by using a 32x32 RGB LED matrix like the one I used in my Light Appliance project (see *N&V's* October 2014 issue).

Bigger is always better, right? **NV**

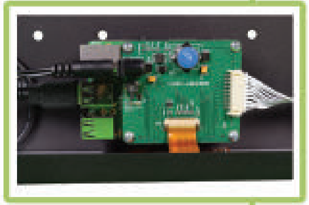
Pi-RAQ

An open source Raspberry Pi Based 1U Rack Mount Internet Appliance With the World's Only 10" x 1" TFT

With the innovation of the Pi-Raq and existing EarthLCD 10x1 display, Rack mount equipment, such as power controllers, audio/video switchers, network hubs and routers, can now have an easily programmable front panel with a color TFT LCD.

Features at a glance:

1- 10" x 1" Color TFT
2- 1024 x 100 pixels
3- 250 Nits
4- 7 Function Tact and Scroll Wheel
5- 9-20V Input Power Supply
6- 10x1 LCD Adapter Board
7- 4 USB Ports



Contact Us: 949-248-2333 Direct
sales@earthlcd.com
www.EarthLCD.com

ELECTRONET

ALL ELECTRONICS CORPORATION
Electronic Parts and Supplies.
www.allelectronics.com
Free 96 page catalog 1-800-826-5432

For the ElectroNet online, go to **www.nutsvolts.com** click ***Electro-Net***

CORIDIUM
Floating point **BASIC** for **ARM** controllers from $5.00
www.coridium.us

INVEST in your BOT!

INVEST in **HiTEC**
12115 Paine Street • Poway, CA 92064 • 858-748-6948 • www.hitecrcd.com

USB Add USB to your next project-- It's easier than you might think! **DLP Design**
USB-FIFO • USB-UART • USB/Microcontroller Boards
RFID Readers • Design/Manufacturing Services Available
Absolutely NO driver software development required!
www.dlpdesign.com

HOBBY ENGINEERING
Kits, Parts and Supplies
www.HobbyEngineering.com

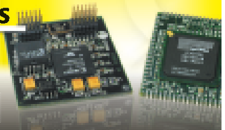
www.servomagazine.com

MOBILE APP NOW AVAILABLE!
SERVO
Download NOW On Your Favorite Mobile Device!
FOR ROBOT BUILDERS

iOS • ANDROID • KINDLE FIRE

PCB,PCBA and More! **Myro** Low cost High Quality
www.myropcb.com
1-888-PCB-MYRO

Ironwood ELECTRONICS
Adapters & Modules
Turn-Key Solutions
Cost Effective High & Low Volume Solutions
RoHS
1-800-404-0204
www.ironwoodelectronics.com



superbrightleds.com
COMPONENT LEDs • LED BULBS • LED ACCENT LIGHTS

Did You Know **Preferred Subscribers get access to all the digital back issues of *Nuts & Volts* for free?**
Call for details
1-877-525-2539

RUNT ROVERS ServoCity.com
Starting at $15.99

GET THE NUTS & VOLTS DISCOUNT!
*Mention or enter coupon code **NVRMZ142** and receive 10% off your order!*
ramsey
www.ramseykits.com
AM/FM Broadcasters • Hobby Kits
Learnng Kits Test Equipment
...AND LOTS OF NEAT STUFF!

Fix Up that Old Radio! Part 2

By J.W. Koebel

It's fun – and easy – to bring your vintage radio back to life!

Last month, we got started repairing a 1937 DeWald Model 618 tube radio. After a short history lesson, we made some basic checks on the radio's important coils and transformers, and discovered that this radio looks like it's in pretty good shape for being 78 years old. However, there's still work to be done! Now, we'll go through the rest of the repair process, a first power-up, and if all goes well we can move on to aligning the radio's tuning dial and enjoying our resurrected antique.

For this step, you'll need some new components. Look at the values printed on the old capacitors, then order new ones in an equal or higher voltage. For electrolytic capacitors, this is usually around 450V, and for the film capacitors 630V should cover all the possible situations. You'll also need some wire snips, needle-nose pliers, solder and a soldering iron, and optionally, a heat gun, hot glue gun, and some bee's wax, and maybe some brown or beige filler material like polymer clay. Why, you ask? More on that in a bit.

If you want to align the radio at the end of the process — this is not strictly necessary, but it makes the stations come in at the correct places on the dial — you could consider using a signal generator with amplitude modulation that has a range from around 450 kHz up through about 2 MHz, or 15 MHz if you want to align the shortwave bands, as well. If you don't have one, don't worry. You can do most of the alignment of the intermediate frequency (IF) and radio frequency (RF) sections with just the existing AM stations you can pick up in your area as test signals. It will come out pretty close.

What are we doing again?

The main cause of age-related problems in an old radio are the capacitors. Vintage caps were made of cardboard, foil, paper, and wax, and just don't hold up well over time — especially not 50+ years. After so long, the materials have just degraded to the point where they no longer work as capacitors.

Dead electrolytic capacitors in the power supply have killed many old radios, and the coupling and bypass capacitors can cause tubes to fail and coils to burn out.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com /index.php?/magazine/article/june2015_Koebel.**

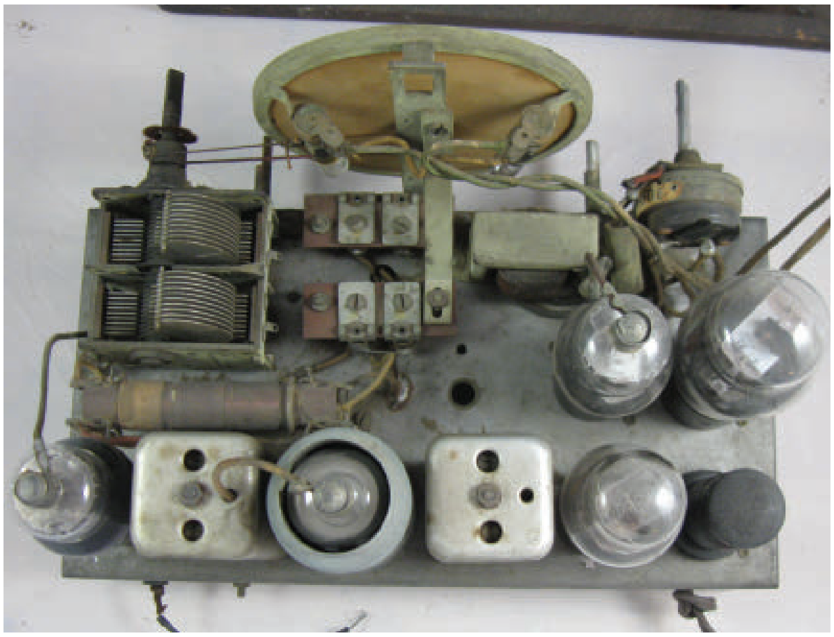


Photo 1. Top-down view of the DeWald 618 chassis.

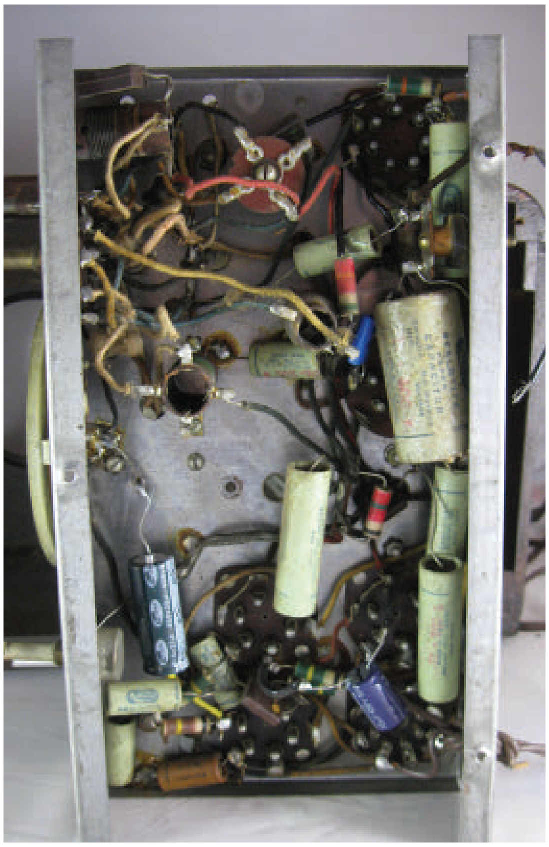


Photo 3. Underside of the chassis. Mostly '30s and early '40s capacitors, with some early 2000 era replacement parts thrown in.

Photo 2. Bad electrolytic capacitors made this transformer burn out.

Take a look at **Photo 2**.

There are a few approaches you can take while going through component replacement. Some people argue it's best to replace only the electrolytic capacitors in the power supply, then one capacitor at a time powering up in between each. This takes quite a while, and with all the other components still original, is pretty risky. I'd personally never recommend such a practice for a radio from the early years, although with something manufactured in the late '50s and into the '60s, you might be able to get away with it.

It looks like someone already tried that approach on a radio a few years ago. Check out **Photo 3**. There are two modern electrolytic and one film capacitor, but the rest are 1930s-1940s replacements.

I like to proceed carefully, one component at a time until all questionable capacitors have been replaced, then power it up and see what happens. By replacing each component one at a time and only lifting one lead at a time, it's easy to make sure you don't mis-wire anything.

Connecting a film capacitor to the wrong place in the circuit won't generally cause anything to burn out (since capacitors block DC current), but can cause the radio not to play, for example, by shunting the radio signal to ground before it gets to the next tube. Just be careful not to connect an electrolytic capacitor backwards.

Photo 4. Capacitor re-stuffing station ready to go.

Be careful working on a vintage radio! There are high voltages present. If you're working with a transformerless radio, make sure to use an isolation transformer as the metal parts might be in direct contact with your AC mains and could cause a serious injury. Use personal protection equipment and good ventilation while soldering. As always, go slowly and be careful!

Under the Chassis

Depending on your goal — a working radio or more of a historically accurate restoration — you can think about some extra steps to prep your new parts before you go through the full replacement. While it can be confidence-inspiring to see a whole set of shiny new film capacitors underneath the chassis, it looks a little out of place among the cloth wiring and hand-wound coils.

For a more period correct look, you can gut the old capacitors and re-stuff them with new replacements. It's the best of both worlds: A period look but the reliability of modern components!

Normally, if you're going to re-stuff the capacitors from your radio, you'd use the existing components which were installed when you found it. Later capacitors weren't good for re-stuffing as the sealing techniques improved and made them much harder to hollow out. This radio's already had some work done, so there aren't enough original components to make a full set.

Luckily, I have a handful of old capacitors I'd kept from previous projects and picked out a few which looked to be about the same color and had the right markings. My DeWald won't look like it's just off the assembly line, but it will still have the same time period look.

Re-stuffing the capacitors is a little time-consuming, but I think it's pretty fun. Grab your tools and supplies: a heat gun, a hot glue gun, clay, wax, the new capacitors, and your old ones to gut. Keep everything handy. You'll want at least one pair of pliers, too, for helping to extract the guts of the old caps. It's a good idea to do this step somewhere that's well ventilated as you'll make some fumes from the heat gun and the wax.

I like to use nitrile gloves while re-stuffing as there's a lot of unknown chemical residue in these old caps. (If you don't have gloves, wash your hands after this step before doing anything else!) Put down some newspaper so the molten wax doesn't go everywhere (**Photo 4**).

Turn the heat gun to moderately high, around 800 degrees F. Hold the capacitor you're re-stuffing by one lead and turn the heat gun on it. Keep rotating around slowly to heat the capacitor until the old wax starts to drip off, taking layers of grime with it and leaving behind mostly clean cardboard. Keep rotating the capacitor and heating with the heat gun until just the faintest whisps of bubbles or smoke starts to come from the ends. Then, secure the body of the capacitor and pull on one of the leads until it pops out.

Use a pen or other small object to push through the body of the capacitor and push the guts out the other side. That's it!

Now, you've got some nasty wax, a couple of loose wire leads, and a bundle of foil densely wound around insulating paper (**Photo 5**).

Photo 5. Empty capacitor tubes and removed guts.

Next, take the hollow

cardboard tube and one of your new components. Axial lead film capacitors work best here, but you can bend the leads of a radial component, and that's just fine. Slide the new cap into the tube, center it, and use a small dab from the heat gun to hold the part in position inside. Let the glue harden for a few seconds, then take your brown or beige polymer clay and push it into the sides, filling up the empty space and forming the end plugs of the capacitor.

Once you've filled in the ends, take the newly re-stuffed capacitor and give it a quick dip once or twice in your molten bee's wax (**Photo 6**). Let it dry for a few seconds and set aside.

You've re-stuffed a capacitor! Repeat for the rest of the caps in your radio, and you'll have a new set (**Photo 7**).

Re-stuffing the electrolytic capacitors is a little harder. Electrolytic capacitors are commonly found in cans above the chassis due to their size back in the day. Some people remove these cans, drill the case open just above where it mounts, extract the guts, and bundle up new electrolytic capacitors inside an insulator before gluing and taping the assembly back together. It's messy and frankly, in my opinion, it's tough to make it look good. Dealing with those hassles isn't an issue for the DeWald, though. It's missing its can capacitor, and there's a set of modern electrolytics instead.

Here, I'm going to fake it a little bit. The radio only needs to really look period correct at first glance, and modern electrolytics are so much smaller than they used to be. I found a slightly larger shell of an old paper capacitor which will fit a pair of 22 µf 450V electrolytic capacitors inside its diameter (back to back), with just a tiny bit of room to spare.

For this one, I gutted the old capacitor, then slid both electrolytics in. Now, there's a paper capacitor with two leads coming out of each end. It's not quite historically accurate, but it looks good at first glance (**Photo 8**).

Photo 6. Dipping a re-stuffed capacitor in new wax to seal it.

Down to Business

Armed with a new set of "vintage" capacitors, it's time to swap out the components. Just cut and replace! Try and get the new capacitors as physically close to the position of the old capacitors as possible. Lead dress — the arrangement of the components and wires — is a big deal in radios this old, and component positions which differ radically from when it was built can cause problems like interference, feedback, and squealing when it's powered up (**Photo 9**).

Sometimes radios like this are built in layers, and you have to remove some top components to get to the ones below them. I use an alligator clip jumper wire on each end of the component I've temporarily bent out of the way to make sure I don't lose my place while working on the deeper layers. Then, I come back and replace the jumper placeholder. I did end up extending some leads for the new "electrolytic" capacitors I installed, using leftover cloth covered wiring from another project to preserve the appearance (**Photo 10**).

While in there, I spot-checked the resistors in the radio. Resistors were made of a molded carbon compound back then, and as they age and absorb moisture from the

Photo 7. The assorted capacitors for this DeWald radio.

Photo 8. Both electrolytic capacitors stuffed into one tube.

Photo 9. Starting to replace the old caps.

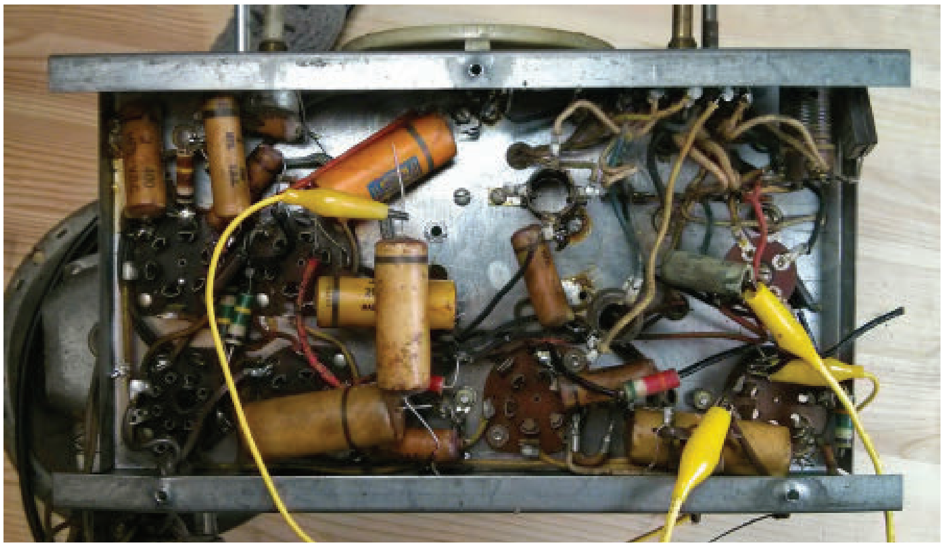


Photo 10. Further along, with jumpers to keep track of parts I've removed to get to the ones below.

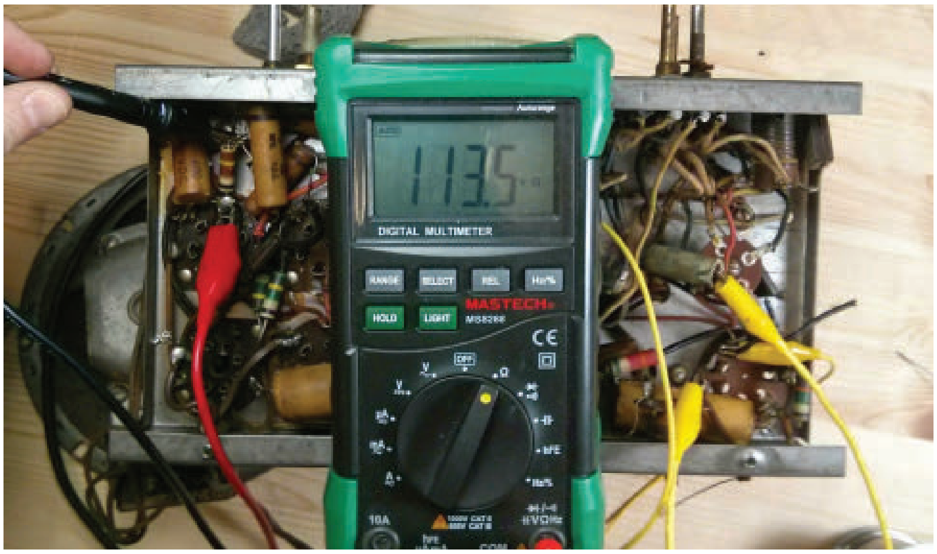


Photo 11. Measuring the drift on a nominal 100K ohm resistor.

atmosphere, they're known to change in value. This doesn't usually keep the radio from working, but it won't be performing at its best with wrong-value resistors. The color codes have the same meaning as they do today, but they're read a little differently: body-tip-dot.

The "body" of the capacitor — the largest color section — is the first digit. The "tip" or color band at one edge of the component is the second digit, and the "dot" — or color band — in the center of the body color is the multiplier. Shown in **Photo 11**, this 100K ohm resistor (brown-black-yellow) is within 20% of its marked value, so doesn't need any further action. I checked a handful of others and they were in spec, so in this case, none needed to be replaced. That's actually pretty rare. I've run into resistors which had drifted to over 100% of their marked value.

You can still get new manufactured carbon composition resistors, although they're a bit more expensive than other types. Visually, they're a different style, looking similar to modern resistors, but those did start turning up in the late '30s. You'd have to replace every one if you wanted a uniform look. It's possible, just a ton of work.

There's not a whole lot more to say about this step. It took about an hour and a half to swap out the components, and now it's ready for the first power-up (**Photo 12**)!

Fingers Crossed

I've double-checked my work to make sure no leads are touching which could cause a short, and reinstalled the tubes and ballast. Now, for the obligatory safety warnings: If you're working with a radio that uses a power transformer, it's fine to connect directly to the mains at this point. For a transformerless radio like the DeWald, however, you **absolutely must use an isolation transformer or you risk a deadly electric shock**.

The radio's metal parts could

Photo 12. Component replacement complete. Looks good!

Photo 13. Isolation transformer for safety.

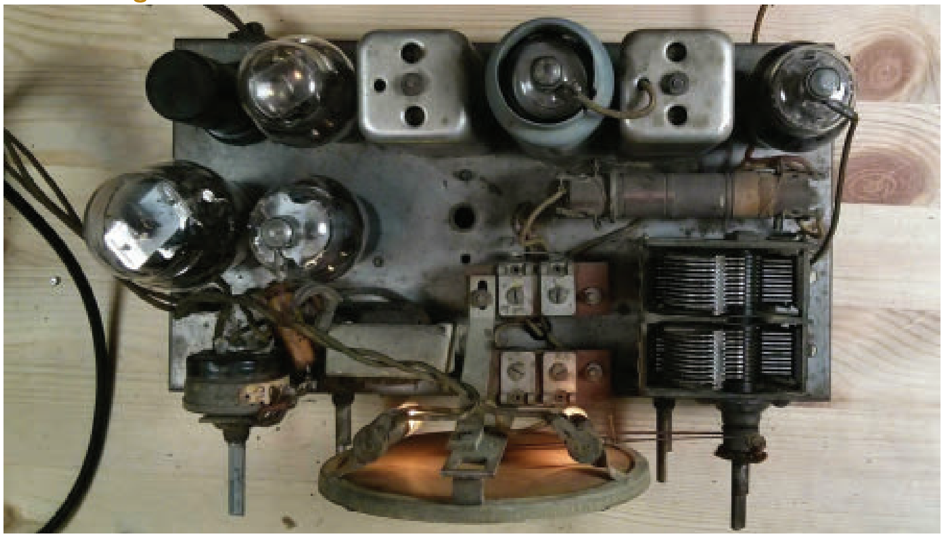


Photo 14. First power-up with the lights on.

become energized at your mains voltage and any contact with an earth ground — like a concrete floor, a cold water pipe, or a piece of grounded test equipment — could cause a short through your body or your gear. Bad news! Don't chance it! Use an isolation transformer and stay safe (**Photo 13**).

After making sure you're safe, flip the switch and hold your breath! This is the part where you'll see smoke if there's a problem (**Photo 14**). On most radios, the switch is on the volume control, so just click it on but don't turn the volume up yet. Otherwise, turn the volume all the way down and flip the power switch separately.

Do the glass tubes start to glow a little inside as their heaters fire up? On a transformerless "series string" radio, if any tube has an open filament or heater, they'll all fail to light and the radio won't get any power at all. So, if you have no power, double-check that the tube filaments aren't open.

Listen for sparking and arcing. If you hear any, power the radio off right away and look underneath to see where it's shorting out and correct that problem.

Do you see or smell any smoke? Likewise, power-off immediately and look to see what went wrong. If it's not obvious, you might need more in-depth diagnostics with help from a place like the Antique Radio Forum (**www.antiqueradios.org**) or the Audio Repair community on Reddit (**www.reddit.com/r/audiorepair**) if this is your first project.

If all you get is a loud 60 or 120 Hz hum that doesn't vary with the volume control, stop. You have an issue with the power supply and need to double-check your filter capacitors. Most of the time the negative terminal of both capacitors will connect to circuit common, but occasionally you'll find a negative-filtered power supply where both positives connect together, but the negative ends connect to different circuit potentials.

One common pitfall is failing to notice that, say, one of two cans on top of the radio chassis is actually insulated and connects to a different negative than the other. Check your connections against the schematic and ask for help if you're still having trouble. Don't run the radio in this condition as you'll likely cause a filter capacitor to explode like a firecracker or damage your transformer.

I flipped the switch and listened, but got absolutely nothing from the speaker. After poking around at various connections and test points, I found the problem pretty quickly: A lead going to the field coil on the back of the speaker had broken! That would pretty much take the power supply out of the circuit.

Fortunately, this was an easy fix, and I was back up and running

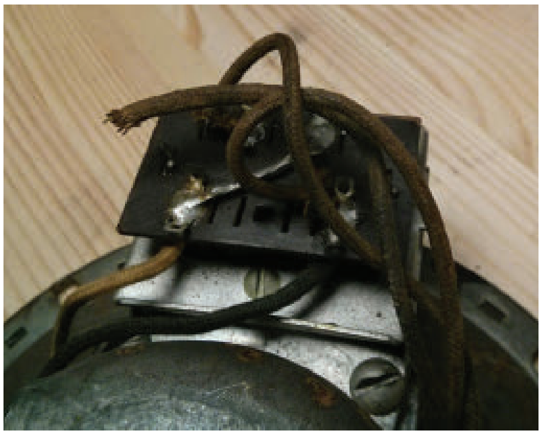


Photo 15. Broken field coil lead.

Photo 16. My signal generator — the LogiMetrics 921A — hooked up to an HP 3585A spectrum analyzer for monitoring.

(**Photo 15**). The next power-up after reconnecting the broken wire started pulling in AM stations on the first try (just like it should) with a very short test lead antenna.

If you're still having trouble getting the radio to make sounds after the component replacement, you'll need to do some more in-depth diagnostics. Double-check that everything is plugged in correctly. If your radio has top caps on the tubes, make sure they're connected. You can carefully tap each of the top caps with the metal end of a screwdriver and listen for a click in the speakers. If you don't hear any clicks, it means there's a problem keeping the signal from passing through the radio properly.

Clicks but still no reception might mean there's a problem with the radio's oscillator or front end keeping the radio signal from being received correctly. Ask for help on a forum if it doesn't seem to be going anywhere.

The DeWald is now pulling in stations, but not quite at the right spot on the dial. It's time to fix that!

Aligning the Radio

Alignment is the process of adjusting the radio's tuned circuits so stations come in loud, clear, and ideally at or near the correct numbers on the dial. While these radios would certainly have peaked up perfectly when they were brand new, age-related changes in the coils and adjustments can make this part a little more difficult. Sometimes the best you can do is "close" but not "perfect." Don't worry too much about it.

If your radio has a set of alignment instructions, go ahead and follow those. There are usually two parts to the alignment: IF alignment, where you adjust the transformers to get signal through the radio as efficiently as possible; and RF alignment, where you adjust the dial tracking and positioning. You can generally adjust both with simple tools like your multimeter and a small screwdriver — or even just your ears!

In the *Rider's Perpetual Troubleshooter* manual for 1937 — a great resource for radio repairs — DeWald provided alignment instructions:

"IF Alignment: Intermediate frequency peaked at 456 KC. Connect test oscillator to grid of 6A7 and chassis. Short circuit stator of front section of variable condenser during this operation. Then, peak IF trimmers for maximum signal."

That's a little bit complicated with the old terminology, but not too bad. To follow those instructions to the letter, you'd set the signal generator to put out a modulated 456 kHz output through a series capacitor to the grid of the 6A7 tube, and connect a jumper across the front tuning capacitor section to short out the incoming signal and ensure you're only getting the test generator's input (**Photo 16**).

Then — with your multimeter hooked up across the speaker's voice coil — adjust the trimmers on the top of the IF transformers for the highest observed voltage. Start out with the last IF transformer closest to the radio's detector, and work your way back to the front towards the 6A7, in this case.

If you don't have a modulated signal generator, you can still do this step by ear. Tune into a loud station, set the volume about mid-way, and listen to the program as you adjust the trimmer for maximum volume and the highest reading on your multimeter. It's that easy!

You'll probably be able to get a tiny bit better results using the full test set, but realistically, you're unlikely to notice a large difference if you just align the IF chain "by ear" versus with a signal generator.

Next up is the RF alignment. DeWald says:

"Remove short from stator of variable condenser. Turn wave band switch to Broadcast. Connect test oscillator to antenna and chassis. Set test oscillator and radio dial to 1500 KC and peak var. Cond. Trimmers for maximum signal. Set test oscillator

You should use a plastic tool to align your radio's IF transformers. Since the high B+ voltage passes through one side of the transformer, one trimmer in each transformer will be at that high voltage. If you touch the grounded metal side of the can, you'll cause a short. If you don't have a plastic tipped screwdriver, make sure to use at least one with an insulated handle and wrap the shaft with electrical tape to make sure it won't short out. Don't use an all-metal screwdriver or you'll get a shock.

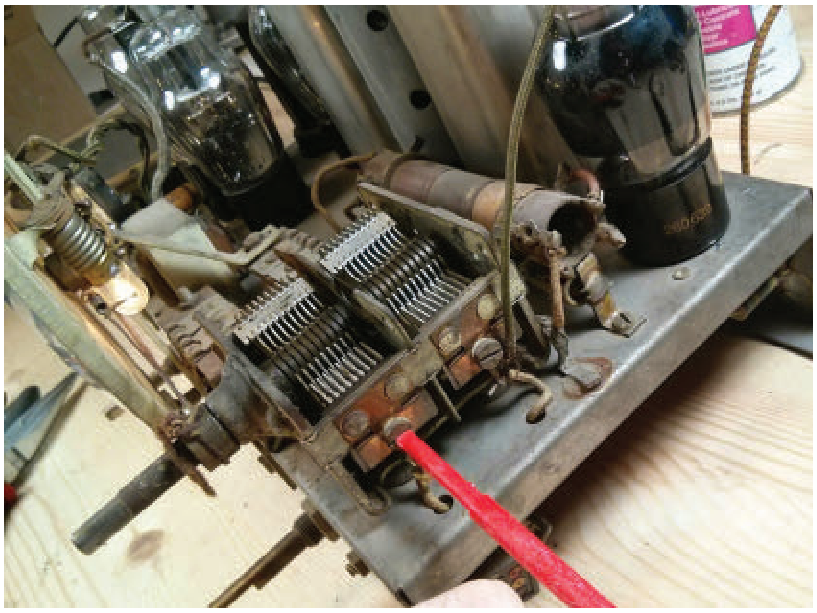


Photo 17. Adjusting the trimmers on the side of the variable capacitor.

[and radio dial] at 600 KC and adjust padder condenser in front of chassis for max signal. During this operation the variable condenser must be rocked. Readjust 1500 KC."

Pretty straightforward. Adjust the high end trimmer to bring the 1500 KC test signal to the correct position on the radio's dial, then adjust the trimmer next to it (which tunes the antenna) for maximum output. Then, move the dial, reset the signal generator, and adjust the second trimmer on the front of the radio for the same. Go back and re-adjust the high end again. This should make the radio's dial track evenly across the whole AM broadcast band.

Don't stress if you can't get it perfect, but it should be much improved (**Photos 17 and 18**).

If you don't have a signal generator, it's still no problem! You'll need to do a little more work, though. Skip the step about shorting out the variable capacitor, find a local AM broadcast station around 600 and 1500 kHz, and tune those stations instead of a test signal. It won't come out perfect, but it will get you pretty close.

At this point, all that's left to do is to enjoy your handiwork for a few minutes, then box it all back up into the case. We've replaced all the components, powered it up, discovered and fixed a broken wire, and aligned the radio for the best possible reception.

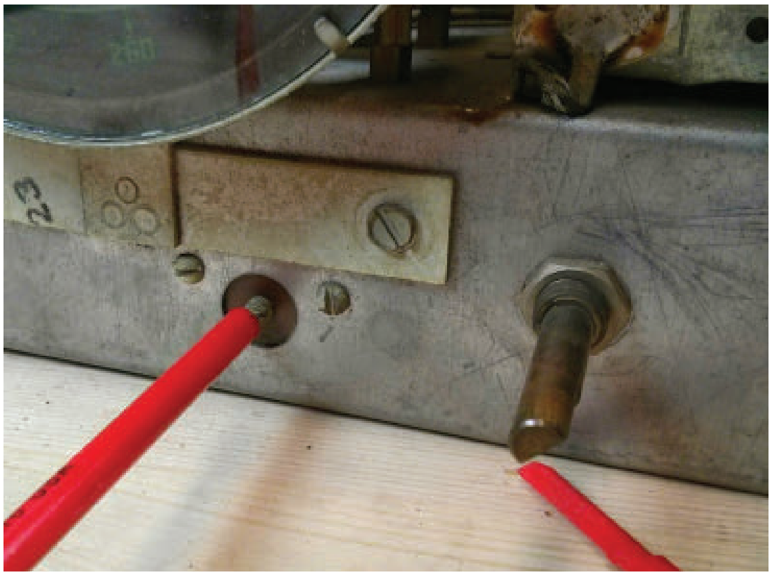


Photo 18. Adjusting the padder trimmer on the front of the radio.

Putting it All Back Together

Put everything back in the cabinet, secure the speaker to its mounting bolts, re-attach any knobs and covers you removed, and reinstall the chassis screws that hold the radio's guts in place. Do one final power-up and tune around to make sure you didn't bump anything in the process. Now, you're all done!

Give the radio a home on your shelf and make sure to turn it on and enjoy it on a regular basis. You've successfully brought a piece of electronics history back to life, and with any luck, you've had fun doing it!

What's Next?

These two articles covered a very easy repair, where everything pretty much went right the first time. In my own personal experience, about three-quarters of the radios I've encountered didn't need any special troubleshooting and started receiving stations the first time they came back online. That's certainly not guaranteed, though.

I'll cover some troubleshooting techniques and some advanced alignment techniques in future articles. Until then, check out the Antique Radio forum or Audio Repair on Reddit for in-depth assistance. There's thousands of friendly active members who love to help — many with experience from when these radios were brand new in the first place.

Until next time, enjoy your radio! NV

Beyond the Arduino 4

Talking to Your AVR Microcontroller

This week, we're going to touch on one of the older more reliable ways to talk to your microcontroller project: using serial communication. You've likely used this with your Arduino board, so let's dive into using it with the microcontroller directly.

All You Want to do is Talk, Talk

Microcontrollers are designed to talk. They want to talk serial, they want to talk I^2C, and they want to talk TWI and SPI. Sometimes they can talk fast, and sometimes they need to talk slow. Some of them can even speak CAN, USB, and Ethernet. But they all want to talk!

Many microcontroller projects need to communicate with modules, other microcontrollers, or the outside world. The way they do this is using the protocols I mentioned in the previous paragraph. Whether you're building a weather station, a home automation system, a remote temperature sensor, or even a simple clock, you will need to use one or more of these protocols.

This month, we'll be covering one of the older and more common serial modes: UART (universal asynchronous receiver/transmitter). If you've been around for as long as I have, you'll remember the days before PCs had USB ports, and most communication took place through an RS-232 serial port — modems, tape drives, joysticks, mice, and even PC-to-PC communications. When we talk "serial" in this article, we're kind of talking about that serial (although technically, what we're working with isn't identical — but that's a discussion for another time).

There are other serial modes that are extremely useful (critical, even) to building more complex microcontroller projects. These include SPI and I^2C, and will be covered in future articles.

Where and When?

Just why is serial communication so useful and worth having an entire article dedicated to it? Personally, I rely on serial in three main areas in my projects — but these are by no means the only or best ways to use serial.

Stamping Out the Bugs

If you've been using an Arduino board for any length of time, I'm certain that you've used the serial library to help with debugging your sketches. Programming a microcontroller is quite different to programming a PC, in that the compiled file is uploaded to a remote MCU. This makes it more difficult to debug, as the code is running on a separate piece of hardware to your IDE (integrated development enviroment). Thankfully, there are debuggers available that allow you to step through your code on your microcontroller, but these don't work in all IDEs. In my Arduino days, I relied a great deal on the *Serial.print()* function to help me troubleshoot my code by sending messages to my PC at select points in the program. As I was learning, I bumped my head many times and *Serial.print()* helped to slow the graying of my hair by speeding up my troubleshooting process!

Even though I now have a debugger that works with my IDE (debuggers aren't, by the way, supported by the Arduino IDE), I still sometimes fall back on the serial port to trace execution of longer programs or of events that happen less frequently.

Streaming Data

Sometimes you want to be able to log or stream data from your microcontroller project for later analysis. For example, I'm currently working on building a reflow oven, and I want to be able to record and graph the temperature profile of my oven on my PC. As serial communication is so ubiquitous, it is incredibly easy to record incoming data on your PC. For one-off analysis, a simple terminal program will do the trick; copied and pasted into a spreadsheet. In the case of my reflow oven, I wrote a short program in Python that collected and graphed the temperatures.

Some really impressive front-ends have been written in a range of languages — Python, Processing, C, and Visual Basic are among the more popular.

By Andrew Retallack
www.crash-bang.com

Terminal Software

In order for your PC to be able to communicate with your microcontroller over the (virtual) serial port, you'll need to use terminal software. This name comes from the old days of dumb terminals that were effectively a screen and a keyboard into another computer. The Arduino IDE comes with simple terminal functionality, which they call the *serial monitor*. The serial monitor — as you've probably found — gives you pretty much all you need to communicate with your MCU.

Atmel Studio doesn't include a terminal application out-the-box, but you can install one as an extension (there are a number of useful add-ons that we can use to extend Atmel Studio's functionality). To install the "Terminal for Atmel Studio" (as it's called), open the Extension Manager (**Figure A**) by choosing it from the **Tools** menu. You can either browse or search for the Terminal for Atmel Studio. Once you've found it, highlight and then click Download. You'll need to sign into Atmel's community called "myAtmel" before completing the download and installation process. The terminal window should be accessible under the **View** menu.

If you prefer not to work in a terminal window within Atmel Studio (and I prefer not to), then any other terminal software should do the trick. I usually use either an open source application called "Tera Term" or PuTTY (refer to the download links in **Resources**).

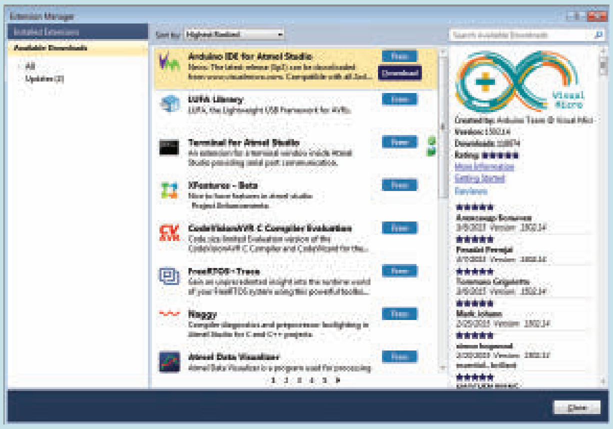


Figure A: Atmel Studio's Extension Manager contains some useful add-ons.

Interacting with Your Project

You may have noticed that your microcontroller didn't come with a keyboard or a screen. As embedded system developers, we need to get creative in how we interact with our projects. In last month's article, we touched on digital and analog inputs, which require some thrifty design if you plan to use them as a way to handle complex interactions with your project. A menu displayed over the serial port can be a great way to interact with your project — whether you choose it as the only way, or in combination with buttons, LEDs, and LCDs.

My first serious microcontroller-based project was the irrigation controller I mentioned in the first article. While I was developing it, I wanted to get the "core" functionality working without spending time on the interface, so I coded a simple serial menu. On startup, a list of configuration options were sent over the serial port and displayed in my terminal software. By pressing certain keys on my PC keyboard (remember serial is a two-way communication), I was able to select menu options, enter irrigation schedules, set the time, and control various other options. I eventually replaced most of the serial menu functionality with an LCD, a couple of pushbuttons, and a rotary encoder, but I left a few "behind-the-scenes" configuration options on the serial menu in case I ever needed them in the future.

Understanding Serial

I'm pretty sure that you want to get a serial connection going as quickly as possible, so may not find the thought of digging into theory very appealing. I hear you, but I think that it's useful to look very briefly at how the UART transfers data.

The ATmega328 has a built-in device called a USART (Universal Synchronous and Asynchronous Receiver/Transmitter) that handles all the complexity of the serial protocol — thankfully! The USART peripheral can actually run in a number of modes to handle synchronous, asynchronous, and SPI communication. In this article, we're focusing on asynchronous serial communication, which is the mode that your computer operates in.

Without getting into the complexities, the very basic difference between synchronous and asynchronous serial communication is that the synchronous mode doesn't only send data — it also sends a clock signal to time (synchronize) the data. The asynchronous mode — without having a clock signal to regulate the timing of the data transmission — needs to send the data at an agreed speed. This is called a baud rate. We're getting ahead of ourselves, though, so let's back-track a bit.

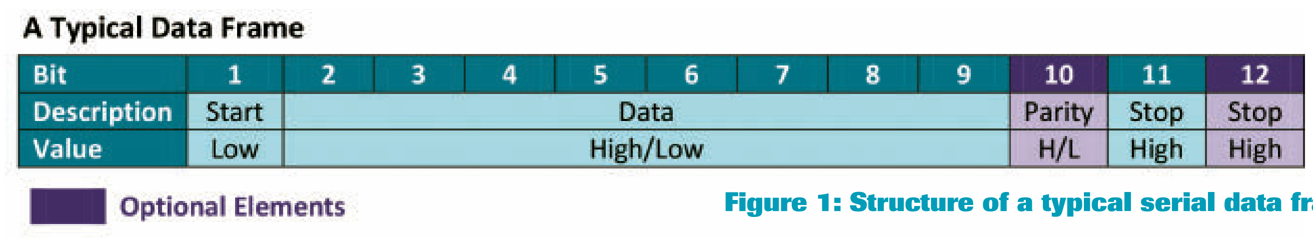
Connect Me Up (Scotty)

The UART uses two pins to transfer data: one for incoming data (RXD, or receive); and one for outgoing data (TXD, or transmit). On the ATmega328P, these are pins 2 and 3, respectively. When you connect two devices, remember that the transmit of each must be connected to the receive of the other — a very common error is to connect the two TXD pins and the two RXD pins. It's no good having two ears listening to each other, and two mouths talking at each other!

Additionally, the devices communicating need to share a common ground. In other words, the GND from both devices needs to be connected. This, simplistically, provides a common reference point for the voltages to be measured off. A high voltage is high relative to ground, and a low voltage is low relative to ground.

Framing Your Data

In order to send data over a UART, you need to frame



A Typical Data Frame

Bit	1	2	3	4	5	6	7	8	9	10	11	12
Description	Start	Data								Parity	Stop	Stop
Value	Low	High/Low								H/L	High	High

🟪 Optional Elements

Figure 1: Structure of a typical serial data frame.

it. (No, not set it up to take a fall, but set it up to succeed!) But, what is a frame?

Many years ago, I used to program in a language called GW-BASIC that ran on the DOS operating system on an IBM PC. When I wanted to open a serial port, I had to pass a whole host of parameters that I didn't fully understand at the time. Does this look at all familiar:

```
OPEN "COM[n]:[speed][,parity][,data][,stop]"
AS [#]filenum
```

The parity, data, and stop parameters in the above command help to define the data frame. **Figure 1** shows how these fit together. Every frame begins with a start bit. When a serial line is in an idle state, it is high (in other words, a "1"). The start bit drops this to a low state ("0") to signal the start of a frame.

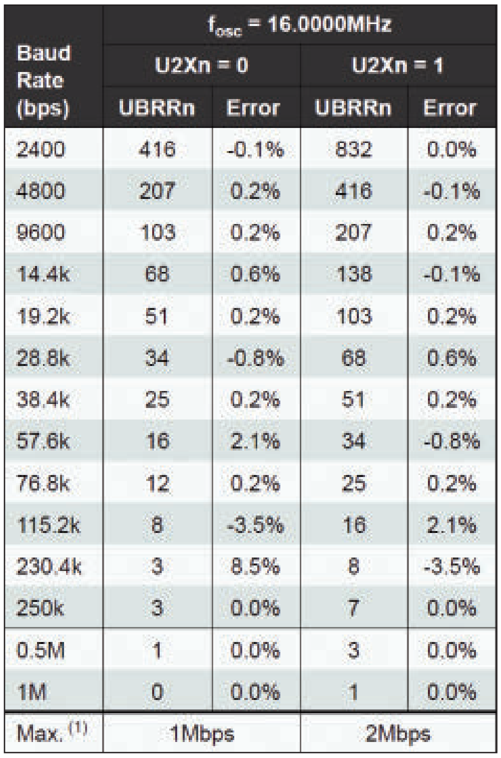
Following the start bit, the data bits are transmitted: low for a 0 and high for a 1. While the data section of the frame can contain between five and nine bits, they commonly carry eight bits — which works out really nicely as eight bits are one byte.

At the end of the data section, you can choose to include a parity bit. A parity bit acts as a very rudimentary form of error checking, helping to flag if the data wasn't correctly transmitted. It isn't used very often, and if used, then the devices need to have a mechanism to handle the re-transmission of failed frames.

Finally, either one or two stop bits are transmitted. These are high, and signal the end of the frame. Usually, one stop bit is enough but if extra processing time is needed, then a second stop bit gives the receiving device a little more breathing space between data frames.

How Fast Should I Go?

The final parameter to set for your asynchronous serial communication is the baud rate. The baud rate is effectively the speed at which data is transferred, and is measured in bits per second. Of course, there are limits to how fast data can reliably be transmitted. A common baud rate is 9600 bps, but speeds of up to 115200 bps are obtainable.



Baud Rate (bps)	f_{osc} = 16.0000MHz			
	U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%
14.4k	68	0.6%	138	-0.1%
19.2k	51	0.2%	103	0.2%
28.8k	34	-0.8%	68	0.6%
38.4k	25	0.2%	51	0.2%
57.6k	16	2.1%	34	-0.8%
76.8k	12	0.2%	25	0.2%
115.2k	8	-3.5%	16	2.1%
230.4k	3	8.5%	8	-3.5%
250k	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%
Max. (1)	1Mbps		2Mbps	

Figure 2: Common baud rates for a 16 MHz crystal.

So, what baud rate should I operate at? The answer is that — just like driving your car — slower is safer, but not always practical. Like many things in the world of microcontrollers, trade-offs exist: In this case, speed vs. error rate. The speed that your microcontroller is running at has a large influence on the baud rates that you can reliably operate at. A faster crystal means the MCU can process at a higher baud rate.

To answer the speed question with empirical data, open up your ATmega328 datasheet and refer to section 20.10 "Examples of Baud Rate Setting." The table in this section shows common baud rates, and the error precent you can expect with common crystal speeds. Take a look at **Figure 2** which shows this information for the 16 MHz crystal that we're using. From the table, you'll see that we can reliably run at baud rates of up to 76800 bps with a less than ± 1% error rate (although I'd probably avoid 57600 bps with its 2.1% error rate). Speed question answered! This table is a great resource, and I refer to it often. As an aside, take note of the "UBRRn" column — we'll need this later.

Where has My PC Serial Port Gone?

Of course, it's very rare to find a PC with a serial port these days — the standard communication port is a USB. The USB standard is quite different to a straight serial one, so we need a mechanism to translate the serial protocol to a USB one. We've already used a USB-to-serial converter in the first article — an FTDI breakout board — so we'll stick with that in this article. Remember that the Arduino UNO, for example, had an additional chip on the board to handle this translation, so this was all transparent to you. When using a USB-to-serial converter, you'll usually install a driver that creates a virtual COM port on your PC. This should be familiar from the Arduino IDE, as you needed to select this virtual COM port when uploading sketches or using the serial monitor.

Opening Channels of Communication

We've now finished the background we needed to cover in order to get stuck in. Last month, we had our first dealings with registers that weren't linked to I/O pins. We used a host of bizarre acronym-like register names to configure our ADC. This month, we're going to need to do something similar as we configure our UART.

We'll need to apply exactly the same practices that we did last month — so pull out the datasheet and turn to section 20.11 which describes the USART registers. If you need to remind yourself of the way that we configure the registers using macros and bit-shifting, then do a quick read-over of last month's article. I've just poured myself a fresh cup of coffee, so here goes!

Configuring the UART

From the background discussions on the various options we can choose when framing our data, you've probably already guessed that we need to let our UART module know how we want to communicate over the serial line. This configuration needs to be performed on both the transmitting device and the receiving device — and you've probably guessed that they need to use the same settings (otherwise, the devices will literally be talking past each other).

The flowchart in **Figure 3** gives an overview of the whole process; let's start by focussing on the red portions.

Step 1: Set the Baud Rate

Setting the baud rate isn't as simple as just passing the chosen value into the microcontroller's register. The baud rate that the UART runs at is linked to the speed you're running your microcontroller at. *You* are the one who needs to take this into account — the microcontroller doesn't! This means you need to calculate the value using:

```
UBRR = F_CPU / 16 / BAUD — 1
```

where:

- UBRR is the value we need to pass to the register
- F_CPU is the MCU speed (e.g., 16000000 if we're running at 16 MHz)
- BAUD is the baud rate you want to run at (e.g., 9600)

Alternatively, you can cheat — perhaps that's too strong a word — short-cut the process and refer to the baud rate tables in the datasheet (see **Figure 2**). Remember the UBRRn column we mentioned earlier? Well, this conveniently gives you the value you need to set the register to. Easy, right? Well, not quite.

To set the baud rate, we need to work with *two* registers: **UBRR0L** and **UBRR0H**. Why two registers? The range of possible UBRR values means that we need to use 12 bits to pass this information onto the microcontroller. Since we're working with an eight-bit microcontroller with eight-bit registers, we can't squeeze 12 bits into the register. So, we need to split the value over two registers (well, one and a half technically, but we leave the remaining four bits unused): the USART baud tate Low and High registers: **UBRR0L** (low) and **UBRR0H** (high).

As an example, let's assume we're running on a 16 MHz crystal and want to set a baud rate of 2400 (very slow, but it works well for this illustration). From the table in **Figure 2**, we need a UBRRn value of 416. Assuming we store this value in the variable *baudRegValue*, the code to set the registers looks like this:

```
unsigned long baudRegValue = 416;
UBRR0H = (unsigned char)(baudRegValue>>8);
UBRR0L = (unsigned char)baudRegValue;
```

You'll recognize the ">>" symbol above as a right-shift — the opposite direction to what we normally use when setting values in registers. To see how this works under the hood, refer to the sidebar, **Shifting Left, Shifting Right**.

Okay, we've set the baud rate. On to the next set of options (and yes, these are simpler).

Step 2: Set the Frame — Data Bits

The UART on the ATmega328P supports between five and nine bits. As I haven't yet needed to particularly optimize the serial communication on my projects, I stick with eight bits as it's easier to work with (and also happens to be the default on most terminal programs).

To configure eight or less data bits, we work with the UCSR0C (USART Control and Status Register C), bits 1

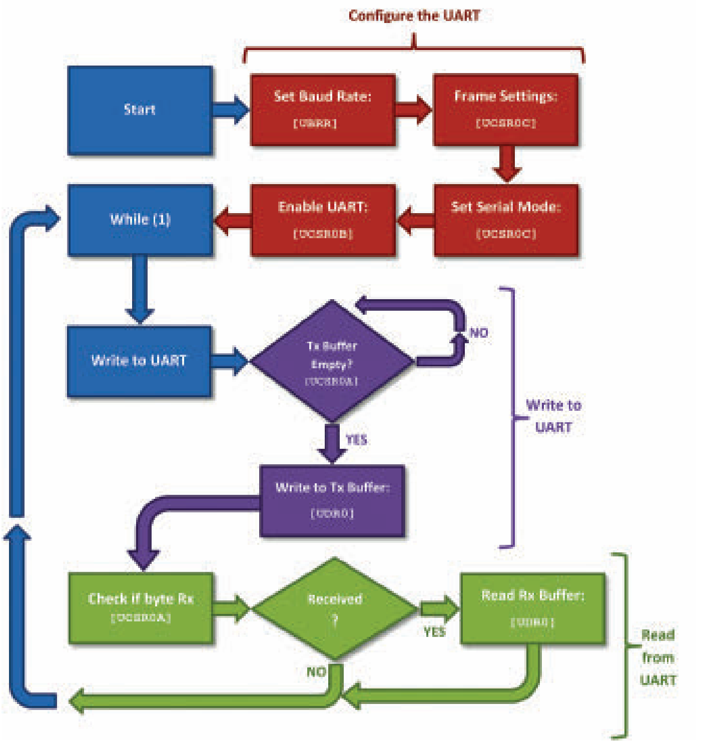


Figure 3: Flowchart outlining basic UART-related processing.

Shifting Left, Shifting Right

In previous articles, we covered and used the left-shift operator (<<) extensively to help with setting or clearing specific bits in registers. We haven't yet looked at the right-shift operator (>>). Of course, it's probably very clear that the right-shift operator must be the opposite of the left-shift operator. That understanding, though, is not very useful in itself. What would you actually use the right-shift for? Certainly not to set bits in a register!

The primary use that I have had for right-shifting is to split a 16-bit word into two eight-bit bytes. If the right-shift is used in combination with type-casting, then you can easily split the word into two bytes. The example below shows this. Assume the variable `wordVar` is a 16-bit word to be split into a `highByte` and a `lowByte`:

```
unsigned long wordVar = 29966;
highByte = (unsigned char)(wordVar >> 8);
        //Extract High Byte
lowByte = (unsigned char)wordVar;
        //Extract Low Byte
```

`wordVar` in binary is `0111010100001110`

Extract the High Byte

To extract the high byte (i.e., the left-most eight bits), we perform a right-shift by eight. This results in a value of `0000000001110101`. Effectively, the original left-most eight bits have pushed the right-most eight bits "out of the variable." The type-casting to an (unsigned char) keeps only the right-most eight bits, resulting in `highByte` having a final value of `01110101`.

Truncate the Low Byte

To extract the low byte (the right-most eight bits), we simply type-cast the value to an (unsigned char). The left-most eight bits are simply truncated, leaving us with a `lowByte` value of `00001110`.

and 2 (named UCSZ00 and UCSZ01). A table in the datasheet under the UCSR0C register description tells us that we need to set both bits to a 1 for an eight-bit frame:

```
    UCSR0C |= ( (1<<UCSZ00) | (1<<UCSZ01) );
```

Step 3: Set the Frame — Parity

From our discussion earlier on parity bits, you've probably guessed that I don't use them much since I haven't needed guaranteed transmission over a serial connection. The parity setting can be found in the UCSR0C register, bits 4 and 5 (UPM00 and UPM01). Instructing the microcontroller to disable parity is the easiest register setting we've yet had to perform — we simply do nothing! That is because the default value is for parity to be disabled (look at the row "Initial Value" on the register table, and compare it to the table of parity bit settings).

If you're writing a library that you'll use in multiple projects, it may be wise to ensure the bits are cleared anyway — particularly if the project communicates with multiple devices with different settings.

Step 4: Set the Frame — Stop Bits

If you're comfortable using one stop bit — which should be fine for most communications, particularly if they aren't that fast — then this is another default value that is left alone. Bit 3 (USBS0) of the UCSR0C register defaults to a 0, and that's exactly what we want for one stop bit. This is starting to get too easy!

Step 5: Select the Serial Mode

We'll be operating in asynchronous mode (the "A" in "UART") which is the default mode for the USART. Therefore, bits 6 and 7 (UMSEL00 and UMSEL01) of the UCSR0C register are left at their default values. We'll change the mode (and the register) in a later article when we start working with SPI.

Final Step: Fire Up the UART

Finally, we need to enable the transmitter and receiver — assuming that we'll be performing two-way communication. We use the USART Control and Status Register B to do this: UCSR0B. Bit 4 (RXEN0) controls the receiver, and bit 3 (TXEN0) controls the transmitter. To enable them, simply write a 1 to each of the bits:

```
    UCSR0B |= ( (1<<TXEN0) | (1<<RXEN0) );
```

That's it — the UART is configured and ready to roll. As with the ADC initialization last month, I normally wrap this up in a function to make for more readable code, and for potential re-use of code.

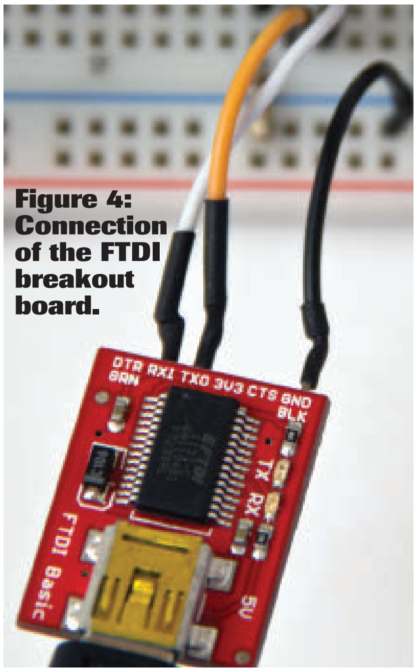


Figure 4: Connection of the FTDI breakout board.

Enough Talking, Let's Start Talking

Let's work through a practical example. I'm sure you want to get the microcontroller talking! **Listing 1** shows the UART-related functions for a simple program that displays the letters of the alphabet on your PC terminal. You can download the full listing for the "Interactive Alphabet" program from the article link.

The program writes the letters of the alphabet out in sequence over the UART, at the same time checking for data received over the UART. If you type a letter into the terminal program, then the microcontroller will read this and reset the counter to start counting from that letter. It's a simple way to show two-way serial communication. Connect an FTDI

Resources

Tera Term Download
http://ttssh2.sourceforge.jp/index.html.en
PuTTY Download
www.chiark.greenend.org.uk/~sgtatham/putty/download.html
Nuts & Volts Forum
http://forum.nutsvolts.com

Parts List

Projects are based on the build from Beyond the Arduino #1 in the March 2015 edition of *Nuts & Volts*. The following additional components are needed for the visual thermostat project:

D1	Green LED 20 mA
D2	Red LED 20 mA
R1, R3	330 ohm Resistor, 0.25 W
TH1	LM35 Temperature Sensor

breakout (or another serial-to-USB converter if you prefer) to your breadboard ATmega328P so that:

- RXI of the FTDI connects to the TXD (pin 3) of the ATmega328P
- TXO of the FTDI connects to the RXD (pin 2) of the ATmega328P
- GND of the FTDI connects to the ground rail of the breadboard

Figure 4 shows the connection of the FTDI breakout module — the only additional component needed for this example. Once your USB-to-serial converter is connected to your PC, the PC should recognize the virtual COM port that is created.

Use this together with the frame settings (eight data bits, no parity, one stop bit) and baud rate (9600 bps) to establish a connection in your terminal software.

Doing the Talking

It's very likely that you're going to want to transmit over the UART in various places in your programs, so it makes sense to include the transmission logic in a separate function that can be called from wherever you need it. Granted the function is pretty straightforward, but it still makes things a whole lot easier. Let's dissect the `UART_writeChar()` function from **Listing 1**, with reference to the purple sections of the flowchart in **Figure 3**. The byte to be transmitted is passed to it as a variable named *data*.

Before we send anything over the serial line, we need to make sure that the transmit buffer is empty (the transmit buffer is a temporary storage area that passes the byte to be sent on to a shift register that actually does the transmission). To do this, we check the UDRE0 bit (bit 5) of the UCSR0A (USART Control and Status A) register. Take a look at the register description in the datasheet and you'll see that the UDRE0 is a 1 if the transmit buffer is empty and ready to receive new data to transmit. So, the line following simply waits until UDRE0 is a 1 — using a mask, of course:

```
    while ( !(UCSR0A & (1<<UDRE0) ) );
```

Once we've been given the thumbs-up to transmit, it's as easy as simply passing the byte to transmit into the UDR0 register — the USART I/O register:

```
    UDR0 = data;
```

If you read the datasheet, you'll see that the UDR0 register is quite an interesting one — it is, in fact, *two* registers that share the same I/O address. If you *write* to the register, the value is placed into the *transmit* data buffer; if, however, you *read* from this register, you'll be

```
void UART_Init(uint16_t baudRate)
{
        uint16_t my_UBRR;

        //Calculate the value of the UBRR
        //register, to set the Baud Rate
        my_UBRR = (F_CPU/16UL/baudRate) - 1;

        //Set the Baud rate
        UBRR0H = (unsigned char)(my_UBRR >> 8);
        //Set the High register
        UBRR0L = (unsigned char)my_UBRR;
        //Set the Low register

        //Stop Bit: 1 - ensure bit is unset
        //(incase it was set before)
        UCSR0C &= ~(1<<USBS0);

        //Data Bits: 8
        UCSR0C |= ( (1<<UCSZ00) | (1<<UCSZ01) );

        //Parity: None - ensure bits unset
        //(incase they were set before)
        UCSR0C &= ~( (1<<UPM00) | (1<<UPM01) );

        //Mode: Ensure Asynchronous Mode (clear
        //bits incase they were set before)
        UCSR0C &= ~( (1<<UMSEL00) | (1<<UMSEL01)
);

        //Enable the Transmitter and Receiver
        UCSR0B |= (1<<TXEN0) | (1<<RXEN0);
}

void UART_writeChar(unsigned char data)
{

    //Wait until transmit buffer empty
    while ( !(UCSR0A & (1<<UDRE0) ) );

    //write char
    UDR0 = data;

}

uint8_t UART_readChar_noWait(void)
{
    // If data has been received read it,
    // otherwise return 0
    if (( UCSR0A & (1 << RXC0 )))
    {
        return UDR0;   // Return the received byte
    }
    else
    {
        return 0;   //nothing received
    }
}
```

Listing 1. Extract of UART-related functions from the "Interactive Alphabet" project.

reading from the *receive* data buffer.

Time to Listen

I have a rather talkative colleague at work, and we often need to remind him to catch his breath and listen to those around him. We'll be doing that ourselves now with the `UART_readChar_nowait()` function.

You're probably wondering why the function name has a "noWait" on the end of it. It's because we need to treat a read slightly differently than a write. In the write function, we waited until the microcontroller was ready to transmit. Once it was, we then fed it the next byte we wanted transmitted. For a read, we don't need to wait for the receiver to be ready to receive. It will automatically receive anything that is sent to it over the serial line. All we need to do is to check whether it has received anything, and if it has then read the received byte. That's what this function does; it's highlighted in the green section of the flowchart in **Figure 3**.

Sometimes, you may actually *want* to wait for an input — for example, if you need input from a user in order for the program to carry on running. Let's say the user needs to set the time for a clock. You'll want to wait until both the hours and minutes have been entered, or your program will continue executing with an invalid time. Just be aware that if you wait for an input and none is received, then your application will appear to hang. It's usually a good idea to implement some kind of timeout when waiting for an input. The following statement checks whether any data has been received by checking the RXC0 bit (bit 7). A 1 indicates that there is data sitting in the receive buffer waiting to be read:

```
    if (( UCSR0A & (1 << RXC0 )))
```

To read the data in the receive buffer, simply access the same UDR0 register; in this case, we return the value from the function:

```
    return UDR0;
```

UCSR0A – USART Control and Status Register A

Bit	7	6	5	4	3	2	1	0
Function	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0

Byte received over UART (RXC0)

Transmit Buffer Empty (UDRE0)

UCSR0B – USART Control and Status Register B

Bit	7	6	5	4	3	2	1	0
Function	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80

Enable Receiver (RXEN0)

Enable Transmitter (TXEN0)

UCSR0C – USART Control and Status Register C

Bit	7	6	5	4	3	2	1	0
Function	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0

USART Mode (UMSEL01, UMSEL00)

Frame: Parity Bit (UPM01, UPM00)

Frame: Stop Bit (USBS0)

Frame: Data Bits (UCSZ01, UCSZ00)

Figure 5: Summary of the USART Status and Control Registers.

You're Conversant

That covers the basics of initializing, writing to, and reading from the UART. The rest of the code in the "Interactive Alphabet" project is pretty self-explanatory, so you should be comfortable following the logic. **Figure 5** summarizes the more complex status and control registers that we've used so far in this article.

Turning Up the Heat

After last month's article dealing with reading analog values, I'm sure you've found things a tad less challenging this month. You are already comfortable working with registers to configure a peripheral, bit-masking is second-nature, and you're probably fairly familiar with basic serial communications from your Arduino board.

So, to stop you from nodding off, let's create a project that brings together everything we've learned so far — or to rephrase, I'd like to challenge *you* to create a project. If that doesn't scare you, you're probably already reading through the specification below and waving your breadboard around; if it comes across a little challenging, then feel free to refer to the "Serial Thermostat" listing that's downloadable from the article link.

As with most embedded projects, there are a number of ways of structuring a program to achieve a goal. The example I've provided is just that — an example. So, don't worry if you've solved the problem in a different way (as long as it isn't using the built-in Arduino commands!). I've commented my example code very heavily in order to link it to the concepts we've covered so far, so it should be easily followed with reference to this and the previous articles in the series.

The Project Specification

We'll be building a simple visual thermostat by combining analog input, digital output, and serial communication. The user will set the desired room temperature by sending a simple command from a terminal program over the serial port to the microcontroller. The project will measure the ambient temperature at regular intervals using an analog temperature sensor, and output the measured temperature on the serial port.

Additionally, LEDs will give a visual indication of the current temperature compared to the entered temperature. If the measured temperature is higher than the target temperature, the red LED will light; if it's lower, the green LED will light. If the measured temperature is within one degree on either side of the target temperature, then both LEDs will light.

The parts list gives the additional components you'll need, and the schematic in **Figure 6** shows how they are connected. The final breadboard layout I used is shown in **Figure 7**.

The Serial Commands

The project accepts two commands over the UART port:

? Displays the help message: "Txx: Set the target temperature to 'xx' Centigrade"

T Waits for two digits to be entered, being the target temperature in Centigrade

Figure 8 shows the output from a typical terminal session, including the temperature outputs from the project, the ? command, and the setting of the target temperature.

Revisiting Temperature Sensors

You may have used the LM35 temperature sensor on your Arduino projects in the past. It effectively contains a variable resistor, with a resistance that changes based on temperature. It has a temperature gradient of 10 mV per degree Centigrade — meaning that for each degree increase in temperature, the output increases by 10 mV. Therefore, unlike the previous article, we need to convert the ADC output (0 to 1023) into a specific voltage in order to determine the temperature. Additionally, the sensor has an offset of two degrees — meaning that it will output a value of 0 mV at two degrees Centigrade. We need to take this into account in our calculations.

Download the datasheet for the sensor that you have, and ensure that you connect the +Vs to the positive power rail, the GND to the ground power rail, and the Vout pin to PC0 on the ATmega328P.

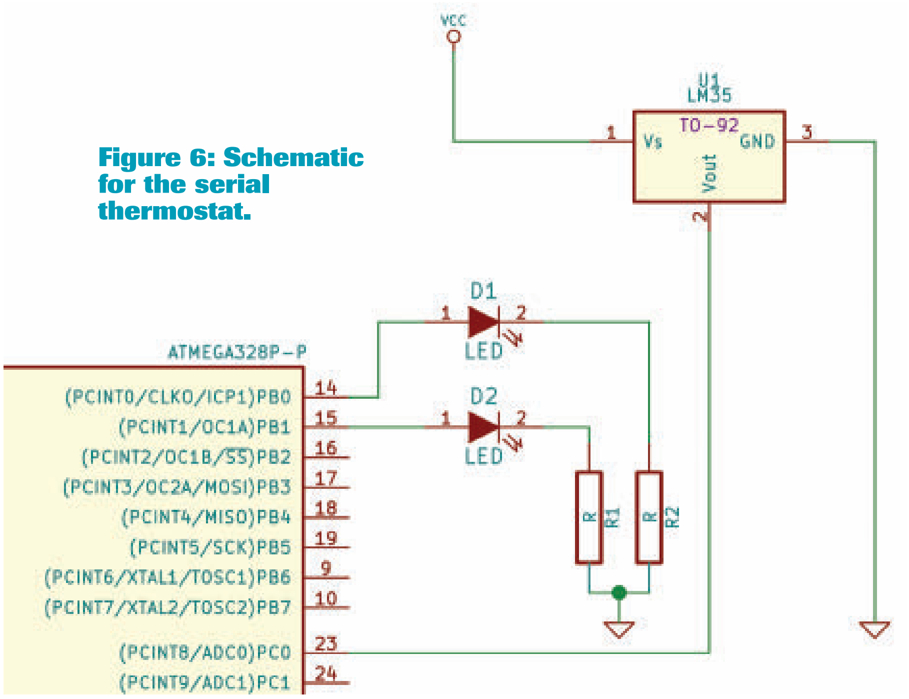


Figure 6: Schematic for the serial thermostat.

Good Luck

Good luck working on this project! If you're anything like me, you'll want to expand it and customize it further. Perhaps, commands to allow a switch between Centigrade and Fahrenheit, or to modify the frequency at which the temperature is measured. You may decide you don't want the LEDs glowing all the time, so simply flash them each time the temperature is sampled. Perhaps pushbuttons could be incorporated.

I'd love to hear about where you went with this project, so please jump onto the *Nuts & Volts* forum (Resources: *N&V* Forum) and share your ideas.

What's Up Next?

If there are specific topics you'd like covered in future editions, please drop us a line and let us know! For now though, I've got a number of topics lined up to give your microcontroller projects increased functionality and improved efficiency. These topics include ways to use interrupts, using internal timers and sleep modes, other useful serial protocols that allow communication with more sophisticated modules (SPI and I^{2}C), using debuggers, and a whole lot more.

Until then, happy communicating, and keep the feedback coming! **NV**

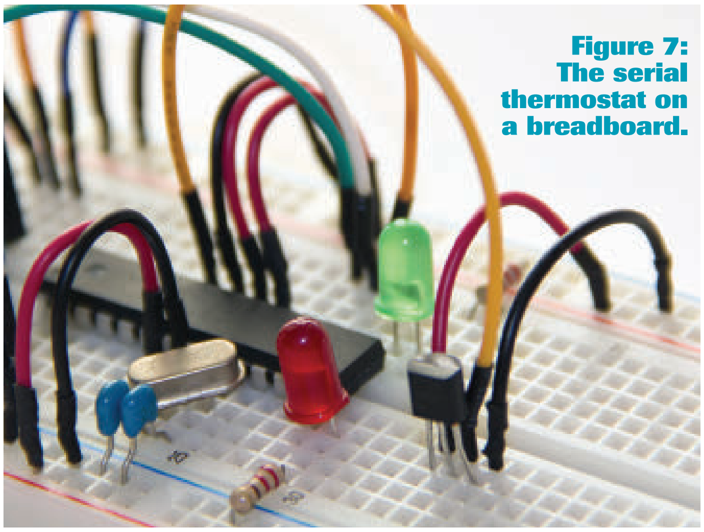
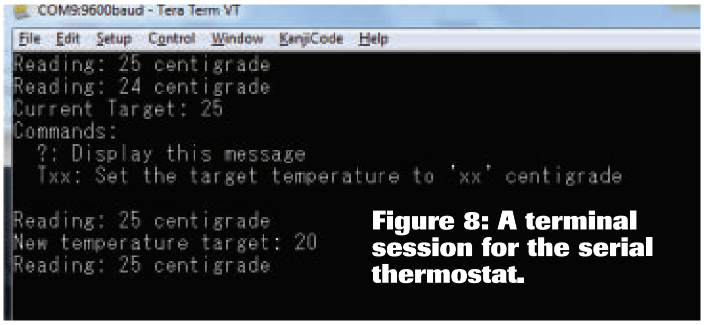


Figure 7: The serial thermostat on a breadboard.


COM9:9600baud - Tera Term VT
File Edit Setup Control Window KanjiCode Help

```
Reading: 25 centigrade
Reading: 24 centigrade
Current Target: 25
Commands:
  ?: Display this message
  Txx: Set the target temperature to 'xx' centigrade

Reading: 25 centigrade
New temperature target: 20
Reading: 25 centigrade
```

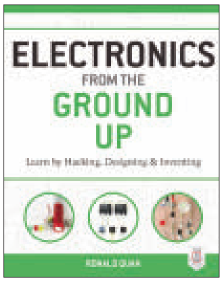
Figure 8: A terminal session for the serial thermostat.

For complete product details, visit our webstore!!

The Nuts & Volts WEBSTORE

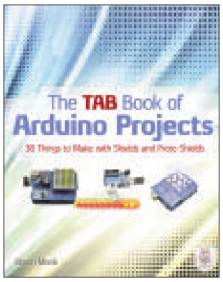
GREAT FOR DIYers!

Electronics from the Ground Up: Learn by Hacking, Designing, and Inventing
by Ronald Quan



Are you fascinated by the power of even the smallest electronic device? Electronics from the Ground Up guides you through step-by-step experiments that reveal how electronic circuits function so you can advance your skills and design custom circuits. You'll work with a range of circuits and signals related to optical emitters and receivers, audio, oscillators, and video.
Paper back 544 pages.
$30.00

The TAB Book of Arduino Projects
by Simon Monk



The ultimate collection of DIY Arduino projects!
In this easy-to-follow book, electronics guru Simon Monk shows you how to create a wide variety of fun and functional gadgets with the Arduino Uno and Leonardo boards. Filled with step-by-step instructions and detailed illustrations, The TAB Book of Arduino Projects: 36 Things to Make with Shields and Proto Shields provides a cost estimate, difficulty level, and list of required components for each project.
$30.00

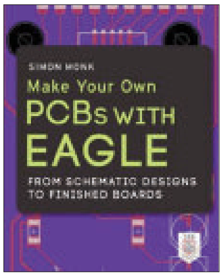
Arduino Projects for Amateur Radio
by Jack Purdum, Dennis Kidder

Boost Your Ham Radio's Capabilities Using Low Cost Arduino Microcontroller Boards

Do you want to increase the functionality and value of your ham radio without spending a lot of money? This book will show you how! *Arduino Projects for Amateur Radio* is filled with step-by-step microcontroller projects you can accomplish on your own — no programming experience necessary.

$30.00

Make Your Own PCBs with EAGLE
by Eric Kleinert

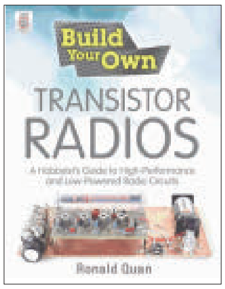


Featuring detailed illustrations and step-by-step instructions, *Make Your Own PCBs with EAGLE* leads you through the process of designing a schematic and transforming it into a PCB layout. You'll then move on to fabrication via the generation of standard Gerber files for submission to a PCB manufacturing service. This practical guide offers an accessible, logical way to learn EAGLE and start producing PCBs as quickly as possible.

$30.00

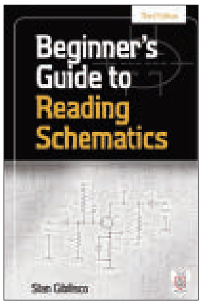
Build Your Own Transistor Radios
by Ronald Quan

A Hobbyist's Guide to High Performance and Low-Powered Radio Circuits



Create sophisticated transistor radios that are inexpensive yet highly efficient. Inside this book, it offers complete projects with detailed schematics and insights on how the radios were designed. Learn how to choose components, construct the different types of radios, and troubleshoot your work.
Paperback, 496 pages
$49.95

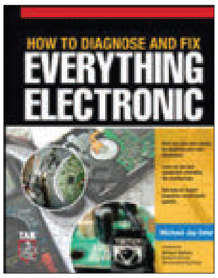
Beginner's Guide to Reading Schematics, 3E
by Stan Gibilisco



Navigate the roadmaps of simple electronic circuits and complex systems with help from an experienced engineer. With all-new art and demo circuits you can build, this hands-on, illustrated guide explains how to understand and create high-precision electronics diagrams. Find out how to identify parts and connections, decipher element ratings, and apply diagram-based information in your own projects.

$25.00

How to Diagnose and Fix Everything Electronic
by Michael Jay Geier



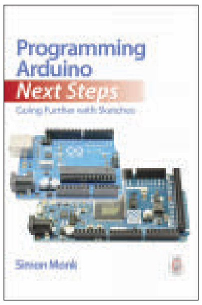
Master the Art of Electronics Repair

In this hands-on guide, a lifelong electronics repair guru shares his tested techniques and invaluable insights. *How to Diagnose and Fix Everything Electronic* shows you how to repair and extend the life of all kinds of solid-state devices, from modern digital gadgetry to cherished analog products of yesteryear.
$24.95

Programming PICs in Basic
by Chuck Hellebuyck

If you wanted to learn how to program microcontrollers, then you've found the right book! Microchip PIC microcontrollers are being designed into electronics throughout the world and none is more popular than the eight-pin version. Now the home hobbyist can create projects with these little microcontrollers using a low cost development tool called the CHIPAXE system and the Basic software language. Chuck Hellebuyck introduces how to use this development setup to build useful projects with an eight-pin PIC12F683 microcontroller. **$14.95**

Programming Arduino Next Steps: Going Further with Sketches
by Simon Monk



In this practical guide, electronics guru Simon Monk takes you under the hood of Arduino and reveals professional programming secrets. Also shows you how to use interrupts, manage memory, program for the Internet, maximize serial communications, perform digital signal processing, and much more. All of the 75+ example sketches featured in the book are available for download. **$20.00**

Order online @ www.store.nutsvolts.com
Or CALL 1-800-783-4624 today!

EDUCATIONAL

Beginners Guide Book Combo.

Beginner's Guide to Embedded C Programming
Beginner's Guide to Embedded C Programming Volume 2
Beginner's Guide to Embedded C Programming Volume 3

Only $85.95 Plus FREE Priority Mail Shipping US Only

DO YOU LOVE RADIOS?

Build Your Own Transistor Radios
Ham and Shortwave Radio for the Electronics Hobbyist
Arduino Projects for Amateur Radio

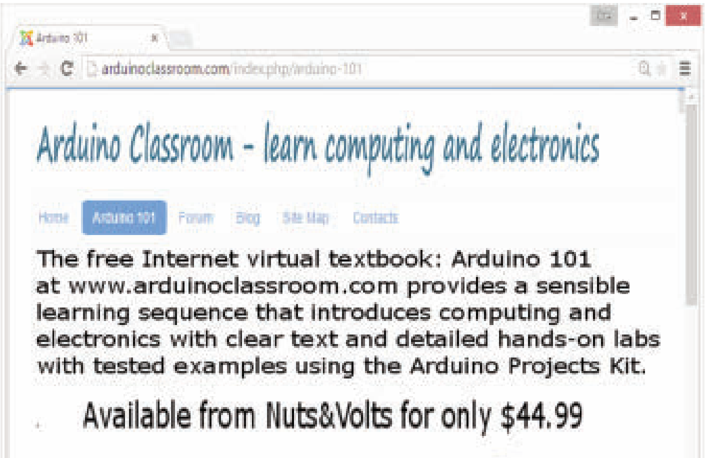
Sale Price $39.95

Sale Price $19.95

Sale Price $23.95

Get 20% off these three books PLUS Free Std. Shipping!

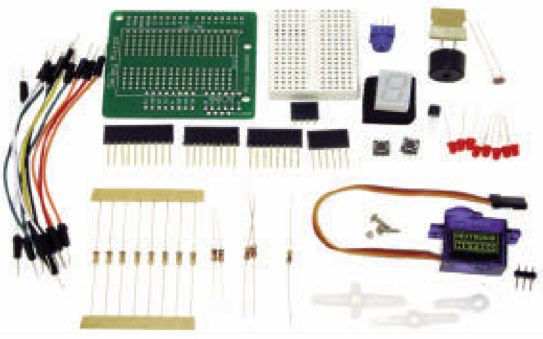
To order call 800 783-4624 or visit: http://store.nutsvolts.com



Arduino Classroom - learn computing and electronics

The free Internet virtual textbook: Arduino 101 at www.arduinoclassroom.com provides a sensible learning sequence that introduces computing and electronics with clear text and detailed hands-on labs with tested examples using the Arduino Projects Kit.

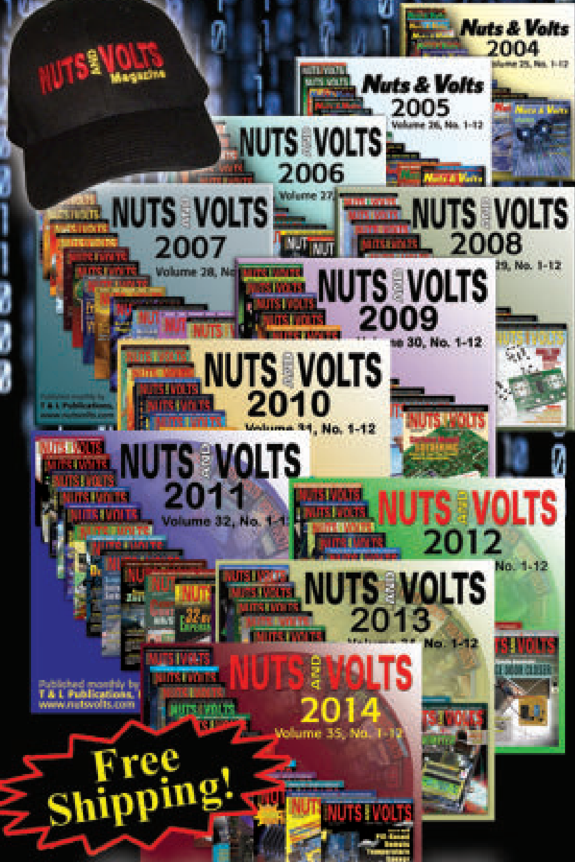
Available from Nuts&Volts for only $44.99



CD-ROM SPECIAL

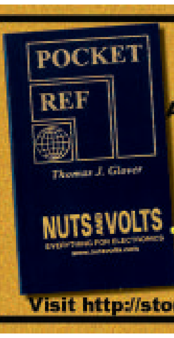
Nuts & Volts 11 CD-ROMs & Hat Special!

That's 132 issues. Complete with supporting code and media files.



Free Shipping!

Only $229.95 or $24.95 each.



The Nuts & Volts Pocket Ref

All the info you need at your fingertips!

This great little book is a concise all-purpose reference featuring hundreds of tables, maps, formulas, constants & conversions.

Only $12.95

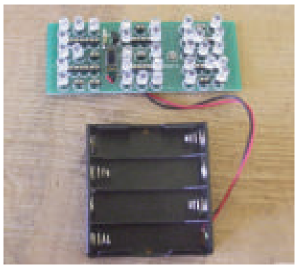
AND it still fits in your shirt pocket!

Visit http://store.nutsvolts.com or call (800) 783-4624

CALL 1-800-783-4624 today!
Order online @ www.nutsvolts.com

PROJECTS

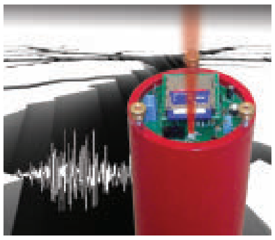
No Nonsense Annunciator Kit



The no nonsense/no microprocessor annunciator is a great little circuit that helps you get your message out without spending too much money. Put two circuits together and you'll have a six letter annunciator!
This kit is also a fun project to refine your soldering skills with its 102 socket pin connection points. WOW, that's a lot of soldering!

Reg $19.95 **Sale Price $14.95**

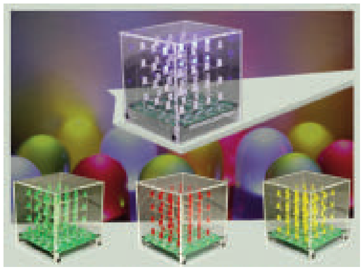
Seismograph Kit



Now you can record your own shaking, rattling, and rolling.
The Poor Man's Seismograph is a great project/device to record any movement in an area where you normally shouldn't have any. The kit includes everything needed to build the seismograph. All you need is your PC, SD card, and to download the free software to view the seismic event graph.

$79.95

3D LED Cube Kit



This kit shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. The preprogrammed microcontroller that includes 29 patterns that will automatically play with a runtime of approximately 6-1/2 minutes.
Colors available: Green, Red, Yellow & Blue

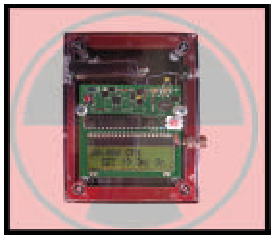
$57.95

Solar Charge Controller Kit 2.0

If you charge batteries using solar panels, then you can't afford not to have them protected from over-charging. This 12 volt/12 amp charge controller is great protection for the money. It is simple to build, ideal for the novice, and no special tools are needed other than a soldering iron and a 9/64" drill!

$27.95

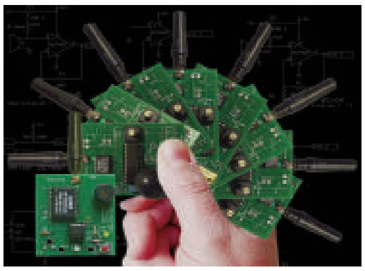
Geiger Counter Kit



This kit is a great project for high school and university students The unit detects and displays levels of radiation, and can detect and display dosage levels as low as one micro-roentgen/hr.
The LND 712 tube in our kit is capable of measuring alpha, beta, and gamma particles.
Partial kits also available.

$159.95

Super Detector Circuit Set

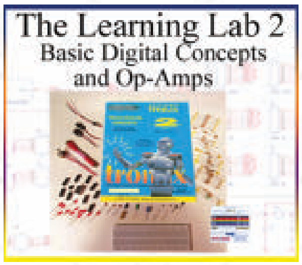


Pick a circuit!
With one PCB you have the option of detecting wirelessly: temperature, vibration, light, sound, motion, normally open switch, normally closed switch, any varying resistor input, voltage input, mA input, and tilt, just to name a few.

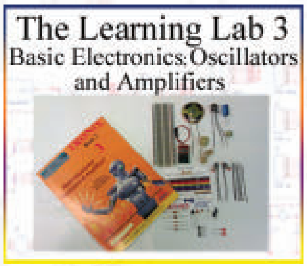
$32.95

FOR BEGINNER GEEKS!

$59.95



$49.95



$39.95

These labs from LF Components show simple and interesting experiments and lessons, all done on a solderless circuit board.

As you do each experiment, you learn how basic components work in a circuit, and continue to build your arsenal of knowledge with each successive experiment.

For more info and lab details, please visit our webstore.

■ BY L. PAUL VERHAGE

CubeSats — Part 4: Some Programs to Launch CubeSats

CubeSats continue to grow in their importance and impact. Although initially dismissed by some as a toy, CubeSats are fulfilling roles carried out by larger satellites from decades past. This month, I'll acquaint readers with some ways CubeSats are reaching space and making personal space exploration a reality.

All Aboard!

With few exceptions, if you've ever wanted to launch a satellite into earth orbit, you paid for the rocket and launch services. This was in many cases a one-time service, so an expensive proposition. Now, organizations are creating opportunities to launch CubeSats and make their access to space more affordable. Let's look at NASA's CubeSat Launch Initiative (or CSLI) for the first example.

The CSLI is an initiative to carry CubeSats as auxiliary payloads on space missions already planned for launch. For a CubeSat to qualify, its mission must meet with NASA's Strategic Plan to expand our knowledge and capabilities in space, increase our understanding of Earth, and meet America's goals in space through the effective use of people and technology.

Through CSLI, NASA is introducing opportunities for greater numbers of post-secondary students to receive a stronger STEM education and employment in STEM-related fields. This means that if you're currently in or about ready to attend college, you have more opportunities to do NASA-related research by getting a CubeSat of your school's design launched into space.

NASA calls these launches of CubeSats ELaNa Missions, or Educational Launch of Nanosatellites. Since ELaNa began launching CubeSats in 2011, there have been eight launches and 36 CubeSat missions deployed, although four CubeSats failed to achieve orbital velocity.

Over 100 school, non-profit, and government institution CubeSats have been selected. Those launched so far include satellites to study the use of smart phones for space communications, measure space weather and radiation, and the deployment of ChipSats — or 1.5" by 1.5" satellites on a chip. Even a high school (Thomas Jefferson High School in Alexandria, VA) has launched a CubeSat onboard an ELaNa mission.

You can read more information on ELaNa at **www.nasa.gov/directorates/heo/home/CubeSats_initiative.html**.

Last year, NASA announced another goal for CSLI: launching 50 CubeSats for 50 states in five years. It's part of a White House Maker Initiative to encourage the increasing population of space enthusiasts to contribute towards space exploration. Making us a space exploration nation — how exciting is that? The program

NASA
ELaNa
NASA·CalPoly
Educational Launch of Nanosatellite
"Launching Education into Space"
"Taking Education to New Heights"

Figure 1. Because of CubeSats, educational institutions are designing and sending experiments into space. *Image courtesy of NASA.*

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/june2015_NearSpace**.

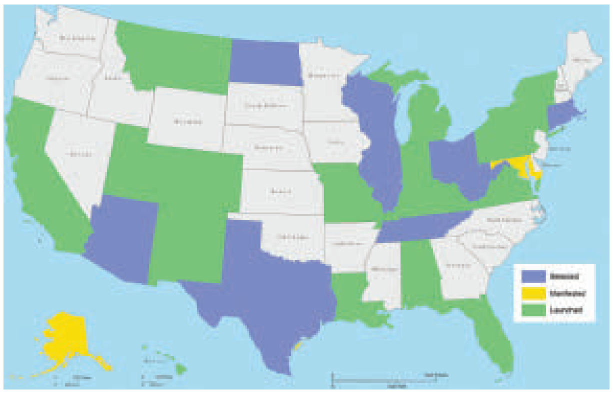


Figure 2. Oops, my state, Idaho was a rookie state without a CubeSat in orbit. Back in February, NASA selected CubeSat proposals and I hope the one Idaho created gets selected. *Image courtesy of NASA.*

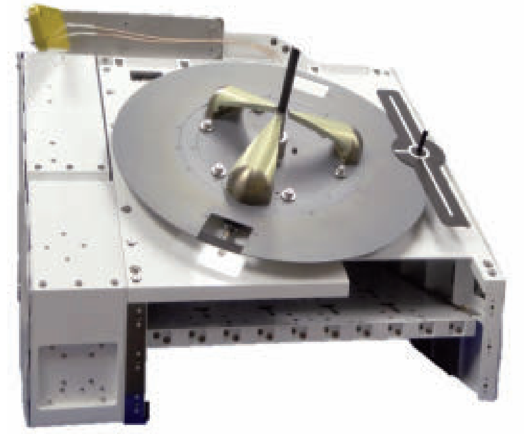


Figure 3. A NanoRacks External Payload Platform, or NREP. The skeletal pyramid on the face of NREP is a standard grappling fixture that permits the robotic arm attached to the Japanese Experiment Module (JEM) to securely grapple the NREP and maneuver it to and from its assigned location on the JEM Exposed Facility and the JEM airlock.

seeks to help the last 20 states which have not launched a government institution or education-based CubeSat get one into space. The map in **Figure 2** shows the states that have had a CubeSat launched, manifested, or selected. Is your state one of them?

Space 4 Everyone

NASA mission launches are not the only way CubeSats can reach space. NanoRacks — a company founded in 2009 — is creating opportunities to send CubeSats into earth orbit. They're dedicated to making space available to everyone, hence their motto: Space 4 Everyone.

One way NanoRacks is increasing public access to space is through their CubeSat deployer onboard ISS. However, the other way I'd like to discuss first is a bit different in that you get your CubeSat back after its mission. It's almost like launching a BalloonSat except you don't need an amateur radio license.

The solution is called the NanoRacks External Payload Platform (NREP) or as the company likes to call it, "The Operating System of Space."

NREP is a platform where NanoRacks can attach customer standardized payload enclosures to support a variety of space-based research. NREP is located on the JEM's Exposed Facility. JEM — along with the H-II Transfer Vehicle (HTV) — are the Japanese Space Agency's (JAXA) contributions to the International Space Station (ISS) — our nation's newest national laboratory.

NanoRacks transports customer payloads inside of space vehicles like the SpaceX Dragon or the Orbital Cygnus to the ISS. Upon arrival, the experiments are unpacked and stored inside of JEM's logistic module. Because of Kibo's airlock, astronauts can attach a payload to the NREP while it's inside Kibo and then use the Kibo robotic arm to transfer the NREP from inside Kibo to its location on the JEM Exposed Facility. There are locations on the Exposed Facility where the NREP is attached to provide power and communications to the attached payload enclosures.

Any experiment attached to NREP must have a CubeSat form factor, but that doesn't mean the experiment must be an actual CubeSat. But hey, why not just send up a barebones CubeSat?

The CubeSat would be exposed to the full vacuum and radiation of earth orbit. Because of the data and power capability of NREP, the CubeSat wouldn't need a radio, ground station, or batteries. Data could be stored onboard the CubeSat or collected via NREP's communication channel and then transmitted to Earth.

With NREP's power and communications ability, there's more room for experiments onboard the CubeSat and it will be cheaper to build. Moreover, as mentioned, you get your "CubeSat" back after its mission!

Kickstarter and Making Space More Available to Everyone

Still need help accessing space for your CubeSat? Well, how about this: NanoRacks will help refine your CubeSat's Kickstarter program. Yep, there have been many teams promoting CubeSat launches with Kickstarter, and NanoRacks was there to ensure they described their CubeSat's mission properly. That way, sponsors were certain they were sponsoring a viable mission.

So, who's used Kickstarter for CubeSats? NanoSatisfi — now called Spire — was the first. On November 19, 2013, they saw their Ardusat-1 and Ardusat-X CubeSats deploy from ISS. A brief search found another publicly-funded CubeSat called SkyCube. Its Kickstarter launch took place on July 14, 2012 and the CubeSat was deployed on February 28, 2014. Planetary Resources is using Kickstarter to support their development and launch of a space telescope for everyone called the Arkyd series 100 satellites. You may have heard that a test satellite of Planetary Resources was launched onboard a SpaceX mission on April 14th this year. However, this was not the space telescope they are developing with the help of sponsors on Kickstarter.

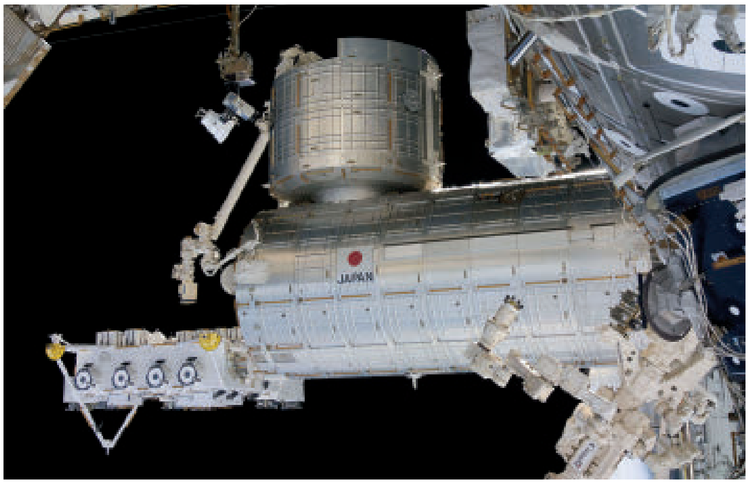


Figure 4. The Japanese Experiment Module, or Kibo. The Exposure Facility is on the left side of this picture; astronauts work inside the pressurized module on the right. Above the pressurized module is the logistics module where payloads are stored. You'll also notice that Kibo has two remote manipulator systems, or robotic arms. *Image courtesy of NASA.*

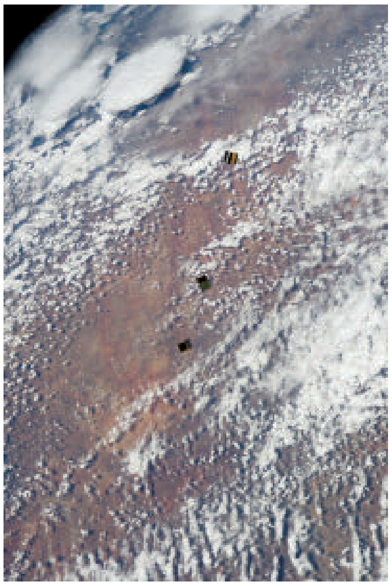


Figure 5. Ardusat-1, Ardusat-X, and Pico Dragon shortly after they were deployed together from the ISS. *Image courtesy of NASA.*

It's interesting the way that a commercial company like NanoRacks uses ISS as a staging point for CubeSats. First, the CubeSats are brought to the space station onboard a government sponsored spaceflight like the Japanese transfer vehicle (HTV) or with a commercial carrier like SpaceX's Dragon or Orbital's Cygnus. Astronauts then transfer the CubeSats to the Kibo module so they can load them into one of two CubeSat deployers.

After loading the CubeSats, the deployer is slid out the airlock where the remote manipulation system can grab and aim the deployer in the desired direction. The CubeSats are then ejected from the deployer using springs and sent on their way.

Over time, their velocity relative to the ISS puts them into an orbit that doesn't intersect the ISS again. These CubeSats typically remain in orbit for at least 90 days before re-entering the atmosphere and safely burning up.

Beyond Earth Orbit

Low earth orbit is not the only destination for CubeSats. Last year, NASA announced a study for the concept of sending CubeSats to far away destinations like Jupiter's moon, Europa. Recall that Europa is Jupiter's bright icy moon that most likely contains a deep liquid ocean. Quite possibly, it's the best location in the solar system besides Earth to have life.

Ten universities (i.e., some professors and their students) are studying the possibility of including CubeSats as auxiliary payloads in a mission called Europa Clipper. The CubeSats could tell a great deal about the gravitational field of Europa by how the distance between the deployed CubeSats and Europa Clipper changes over time.

Knowing the structure of Europa's gravity field gives clues about the moon's internal structure and the possible presence of an ocean of liquid water. Radiation measurements conducted by the deployed CubeSats could indicate the presence and behavior of a magnetic field and therefore about possible conducting fluid residing within the moon. So, keep your ears opened. Some time this summer we may hear more news about carrying CubeSats to exotic destinations like this.

I hope you've become a little more acquainted with CubeSats after reading this series of articles. Next time, I'll introduce readers to my new project: a model CubeSat designed for the high school class setting. With luck, I'll have the opportunity to test it in my classroom sometime next year.

Onwards and Upwards,
Your near space guide **NV**

NEW PRODUCTS

Continued from page 21

in order to create separate files for both the embedded software and mobile app.

• Image Upload and Base64 Encoding enables the developer to easily add images (such as brand marks, icons, and buttons) to an Atmosphere project from a local computer; Base64 Encoding then adds such images directly to the project for easier offline use.

• More sample projects. When useful to the beginning developer, Anaren's "sample" projects can be used, modified, tested, and deployed to the application, saving time.

Miscellaneous Improvements:

• Improved compatibility between the Atmosphere developer tool and touch-enabled laptops like Surface Pro.

• Improved compatibility with iOS tablets.

• Added compression to network transmission for reduced file sizes.

• Support of Windows mobile app compilation.

• Enhanced developer tool compatibility with DPI setting changes.

• Improved new-user account registration process.

For more information, contact:
Anaren, Inc.
Web: **www.anaren.com**

If you have a new product that you would like us to run in our New Products section, please email a short description (300-500 words) and a photo of your product to:

newproducts@nutsvolts.com

CLASSIFIEDS

SURPLUS

SURPLUS ELECTRONIC PARTS & ACCESSORIES

Over 20,000 Items In Stock

Belts, Cables, Connectors, Fans
Hardware, LEDs, Motors, Potentiometers
Relays, Semiconductors, Service Manuals, Speakers
Switches, Test Equipment, Tools, VCR Parts

Surplus Material Components
SMC ELECTRONICS
www.smcelectronics.com

No Minimum Order.
Credit Cards and PAYPAL Accepted.
Flat $4.95 per order USA Shipping.

KITS/PLANS

QKITS LTD
sales@qkits.com
1 888 GO 4 KITS

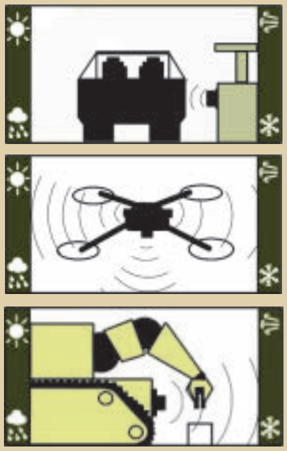
Arduino • Raspberry Pi
Power Supplies
MG Chemicals
RFID

$8.50 flat rate shipping

Visit us at:
www.qkits.com

ROBOTICS

Need sensors?



www.maxbotix.com

CONTROLLERS

Join The
INTERNET of THINGS
REVOLUTION

TRI SUPER PLCs
Powerful & Easy Ladder
+BASIC Programming
Ethernet integrated
MODBUS TCP/IP
DI/Os & AI/Os integrated

tel : 1 877 TRI-PLCS
web : www.triplc.com/nv.htm

TRIANGLE RESEARCH INTERNATIONAL

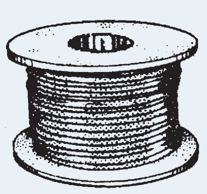
Like to build robots? Then, you need a subscription to *SERVO*!

www.servomagazine.com

HARDWARE WANTED

DEC EQUIPMENT WANTED!!!
Digital Equipment Corp.
and compatibles.
Buy - Sell - Trade
CALL KEYWAYS 937-847-2300
or **email buyer@keyways.com**

LIGHTING



ANAHEIM WIRE, INC. Master distributor of electrical and electronic wire and cable since 1973. Items available from stock: Hook up wire, Shrink tubing, Cable ties, Connectors, etc. Wire cut & strip to specs, twisting, striping. If interested, please call **1-800-626-7540**, FAX: 714-563-8309. Visa/MC/Amex. **See us on the Internet**: www.anaheimwire.com or email: info@anaheimwire.com.

SERVICES

Sandtronics

Sandpiper Electronics Services LLC

"design of custom electronics instrumentation"

Circuit design & drawings. PCB layout, fabrication & population
Prototype development for research, industry and inventors.
39 years experience in electronics development for research & flight applications , laser systems & laboratory research

FREE no obligation consultations

www.sandtronics.com

AUDIO/VIDEO

PARTS EXPRESS
YOUR ELECTRONICS CONNECTION

OVER 18,000 ELECTRONIC PARTS IN STOCK

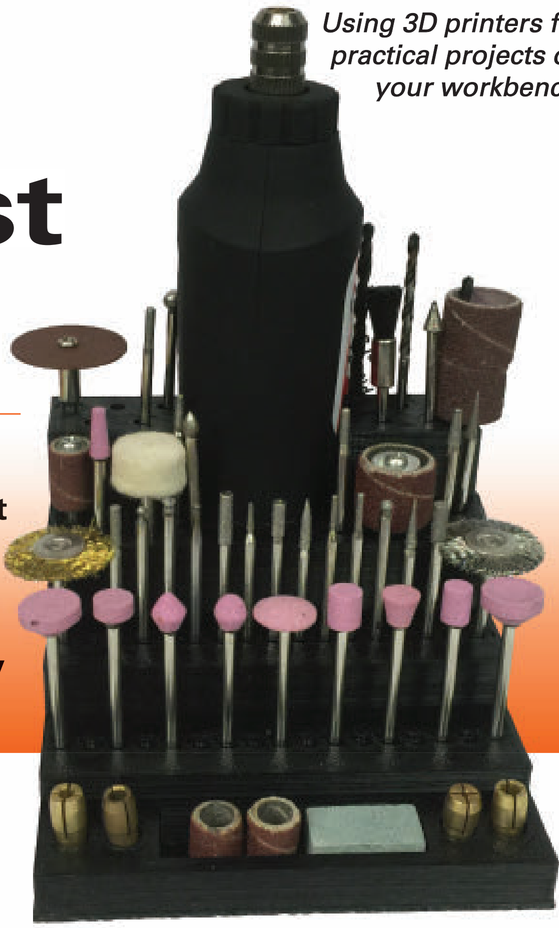
Call or visit us online today to receive your **FREE** copy of our 2015 catalog!

parts-express.com/nuts
1-800-338-0531

■ BY CHUCK HELLEBUYCK

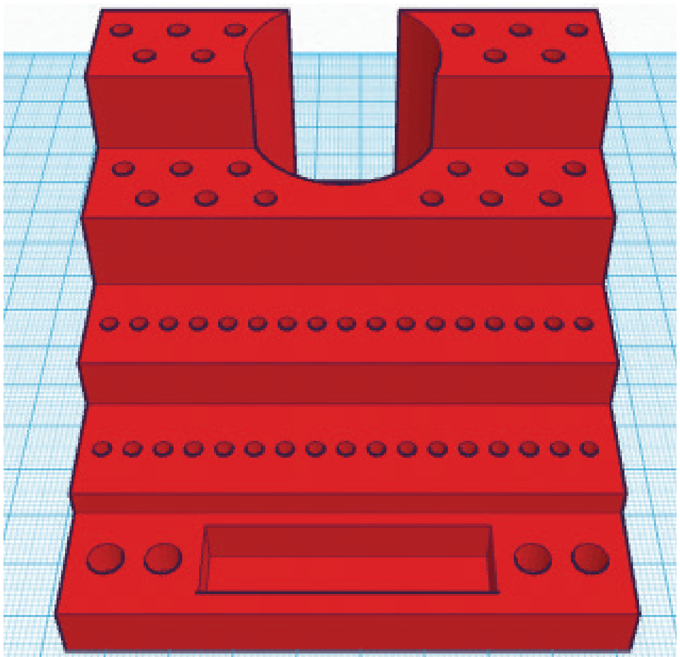
Using 3D printers for practical projects on your workbench.

Tool Stand for a Low Cost Rotary Tool



Organizing my bench is always a challenge. I like to keep everything visible so I can see and use it without digging around. That requirement can easily make my bench messy as many tools are scattered about. So, when I can find a way to organize and also display my tools, I jump at the opportunity.

Recently, I purchased a low cost rotary instrument that is similar to a Dremel style tool, but far less powerful and a whole lot cheaper. I just needed it to drill small holes or grind off some plastic from a 3D print. The tool came with a whole bunch of tiny interchangeable bits. If I laid them all out on my bench top, there would be no more room for anything else. Since they were all round, a tiny bump to the bench would find them rolling on the floor. So, with the help of Tinkercad software, I set out to create a 3D printable tool stand customized for this specific device.



■ FIGURE 1. Tinkercad design

The design started out simple enough. Just a block with holes for holding the tool bits. However, after counting the number of bits, I found I would need a lot of these blocks, so I started stacking them like stairs. Before long, I had a back section that was half the height of the rotary tool — just perfect for a large slot to hold the tool when idle.

The bits came in two basic diameters: 0.090 inches and 0.120 inches. Then, there were different loose collets of various sizes for inserting into the rotary tool chuck. I located the 0.090 holes along the bottom rows, and the 0.120 holes along the top. In the front block, I added a tray for loose bits and larger holes to fit the collets.

Construction

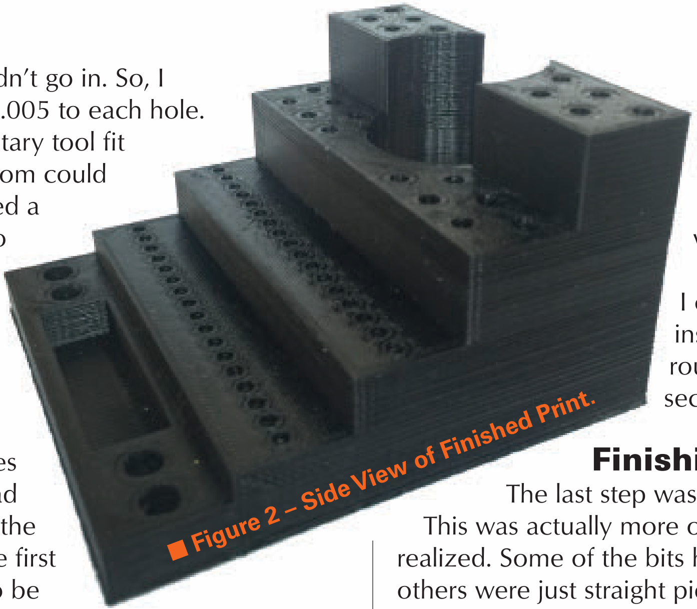
I don't think you can ever get a design right the first time. Not a problem if you have your own 3D printer vs. sending the design out to be printed. The print took about eight hours to complete, then I was able to test it out. It didn't take long to find that the holes needed to be enlarged a little because as the printer builds the design layer by layer, it leaves ribs of plastic at each layer. The center of that plastic filament is at the dimension, however, it gets squished down a bit, leaving a little plastic inside where there should be an open hole. It made some parts

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/june2015_Practical3DPrinting**.

fit tight, but others just wouldn't go in. So, I changed the design to add 0.005 to each hole.

The round slot for the rotary tool fit reasonably well, but the bottom could slide out the back. So, I added a wall to the back of the slot to hold the tool in place. I also reduced the diameter a little for a slightly better fit.

Another eight hours of printing, and the design was perfect. The form measured out as expected and the holes were now a perfect size. I had used washable stick glue on the heated bed to hold down the first layer of plastic, so this had to be washed off from the bottom. *Getting that first layer to stick is critical for any 3D print. Some people use blue painter's tape; some use hair spray; and some use Kapton tape, while others just rely on the heated bed.*


■ Figure 2 – Side View of Finished Print.

Printing Details

I printed my new tool rack in black ABS plastic, which is the same plastic used to make LEGO blocks. The design was printed at a 0.2 mm layer height and a 30% fill. This is the density of the plastic inside. *A 100% fill would be a solid piece of plastic.* I also had it printed with thick shells.

Shells are the outlines around any edge — such as the holes and the outside walls. Thick shells on my Da Vinci printer are three layers of solid plastic around every edge. This gave it a strong structure around all the holes.

Resources

Check out my website and blog:
www.elproducts.com

YouTube Channel:
www.youtube.com/user/beginnerelectronics

My 3D designs:
www.thingiverse.com/elproducts/designs

Tinkercad:
www.tinkercad.com

Da Vinci 3D:
us.xyzPrinting.com

Da Vinci Forums:
www.solidforum.com
www.voltivo.com

Rotary Tool
www.harborfreight.com/80-piece-rotary-tool-kit-97626.html

The base was plenty long enough to balance the weight of the rotary tool at the back. Plus, when all the bits are installed, it will have more weight in the front.

The tray turned out great as I could put a sharpening stone inside for the drill bits and a few round sanding disks. Overall, the second print was perfect!

Finishing Up

The last step was to install the bits and pieces. This was actually more of a puzzle challenge than I realized. Some of the bits had large diameter tops, while others were just straight pieces. I arranged them as best I could and placed them in a way that it was easy to grab them. The stacking also allowed me to see everything right in front of me without a big mess in my work area.

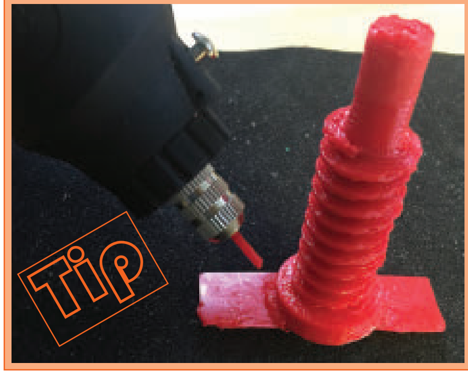
The tool now sits on my bench on the opposite side of the soldering iron. I've used it to cut circuit board traces; grind down sections of plastic on 3D prints; I've even used it for plastic welding.

Having the rotary tool and all its bits easily accessible and organized was a great addition to my workbench. I doubt I could find anything designed perfectly for my needs, even if I was willing to pay a lot for it. The amount of plastic to build the unit cost about $5. So, between the two versions I spent a total of $10. *The first print was actually still usable for other tools such as small screwdrivers and drill bits, so that wasn't a waste either.*

Now, I have to pick the next tool to work on to help clean up my bench. I think I'll attack the soldering iron and all its bits and pieces that seem to make a mess on that side of my work area. **NV**

Friction Welding with a Rotory Tool

I tried a technique where you put a small piece of filament in the tool instead of a bit. The rotation of the filament, when pressed against other plastic, will cause friction and enough heat to melt the plastic. Then, you can use that melted plastic to weld two pieces together. *It worked really well.*



■ BY FRED EADY

Adding a microSD Card Demystified

Microchip is known for its low cost, low pin count microcontrollers. Lifting the covers on various commercially available gadgets will reveal that a resource-stingy baseline (PIC16xxxx) or midrange (PIC18xxxx) PIC microcontroller is in charge. Usually, the ruling PIC is small in stature and huge in importance as far as the operation of the gadget is concerned. Although many PIC based commercial applications can be realized with small amounts of supporting volatile and nonvolatile memory resources, some applications move enough information to require the installation of a "silicon safe" to store the large amounts of important data that the application is required to process and protect. The new line of enhanced midrange PICs are enabled with SPI and I²C interfaces, as well as an upgraded compliment of I/O and analog-to-digital pins. If the compute power of an enhanced midrange PIC is sufficient to handle the I/O traffic load, so be it. However, if your application feels the need for speed, requires a large amount of I/O activity, and needs to store and shield critical data, you may find yourself designing the application around one of Microchip's 32-bit PICs. You may also find yourself designing in and mounting some sort of nonvolatile storage device capable of swallowing and regurgitating the large amounts of data your application will be responsible for.

Today's nonvolatile memory ICs are getting denser and denser. It's not uncommon to find megabytes of nonvolatile storage crammed into an eight-pin memory IC. Despite the arrival of Beagles, Boxers, Raspberry and Banana Pis, the good old PC is still holding the high ground. Many times, data captured by a microcontroller based device ends up being analyzed on a PC. Sometimes it's easier or even necessary to move data between a microcontroller and PC via a physical storage device.

To make that happen, the physical storage device must be able to be accessed by both the microcontroller and the receiving computer. That is the reason that most all of the new PCs sport an SD card slot. As of this writing, you can't plug an eight-pin memory IC into a computer's SD memory card slot. However, you can plug a PC-friendly SD card into a microcontroller card socket.

Tooling Up

I'll be honest with you. There have been times I committed to a project just to be able to use the fancy compilation and source editor tools the project required. I have a soft spot for source code editors that employ tricky features that help the author write well laid out and organized source code with the least effort. IDEs (integrated development environments) are also a favorite tool. When I'm writing Beagle Bone and Raspberry Pi apps, I enjoy manipulating the Eclipse IDE as much as I do writing the code. When it comes to 32-bit PIC code development, I prefer Microchip's MPLAB X.

It took a while, but I finally weaned myself away from the good old standard MPLAB. I can recall a time when I was pushing out of the microcontroller assembler shell into the world of microcontrollers programmed using the C language. It's a similar feeling moving from MPLAB to MPLAB X. I was there for the very first betas and initial production release of MPLAB X. This month, I'm writing a Design Cycle piece about the latest beta version of MPLAB X (3.00) that will be used to craft a PIC32 microSD card driver using the latest XC32 C compiler (1.34).

The really cool part of this is we're using the latest and greatest Microchip tools and basing our code on proven legacy microSD card drivers. To follow along with this month's project, download and install the MLA (Microchip Libraries for Applications) version entitled *microchip_solutions_v2013-06-15*. The project skeleton for the PIC32MX we will be discussing can be found in the MLA's *MDD File System-SD Card* directory.

Hardware Setup

The MLAs are working implementations of code that are designed to be modified by the programmer for his or her particular needs. The hardware behind the MLAs is usually one of Microchip's development tools such as the Explorer 16 series of development boards or USB and

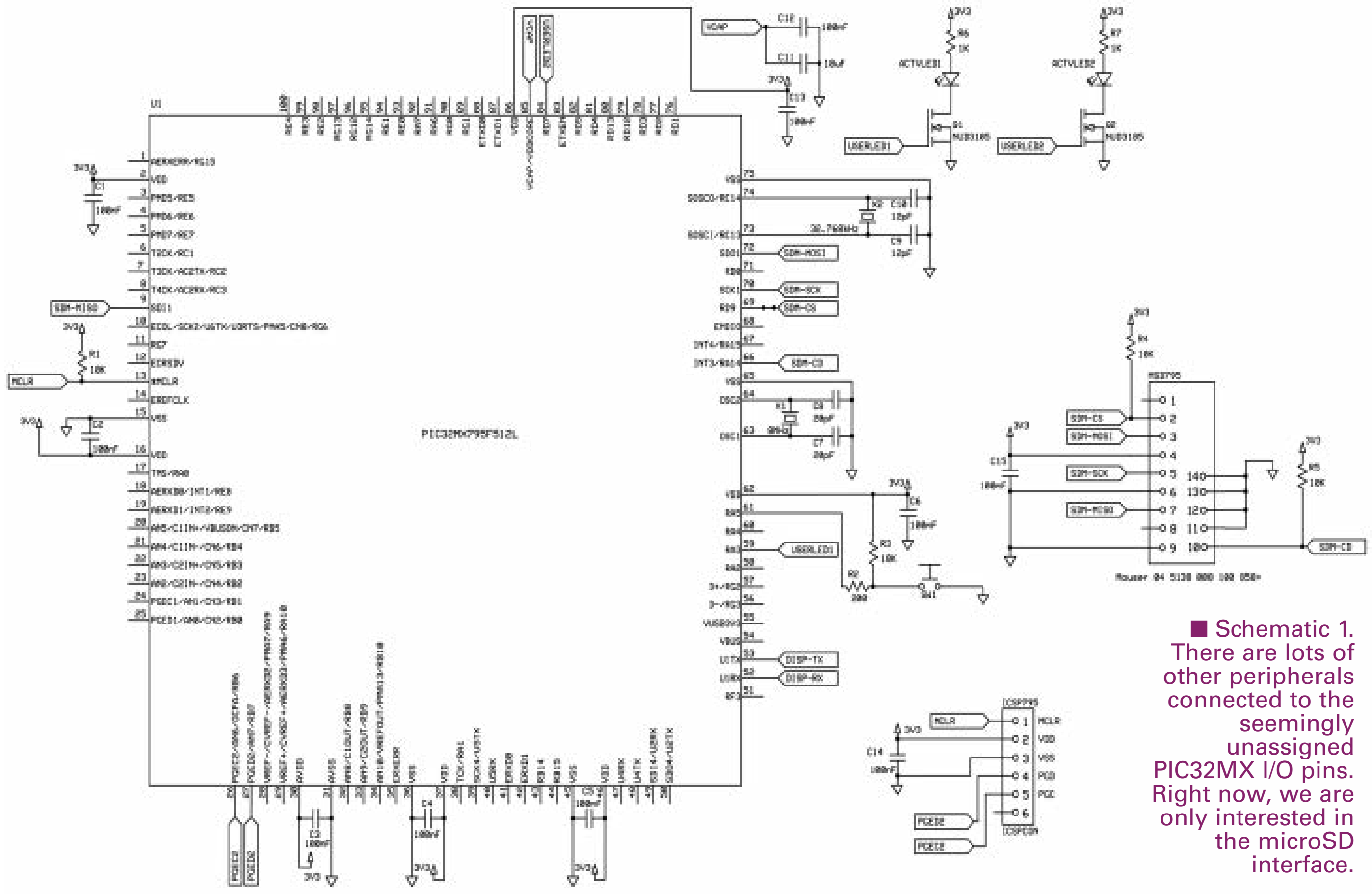
Ethernet starter kits. We will be writing code for a "user defined" hardware platform. The 32-bit hardware scheme we will be targeting is graphically depicted in **Schematic 1**. For the sake of simplicity, I've only drawn up the circuitry that we will be directly affecting, which happens to be the microSD interface. Since communicating with the microSD is our ultimate goal, let's examine that piece of the system first.

Note that the microSD connector is an off-the-shelf part that is equipped with a card detect switch. There is no *write protect* or *write enable* signal associated with the microSD connector we will be working with. However, if we examine the microSD driver source code, we find references to *card detect* and *write enable* signals. Here's the code excerpt found in the MLA project's *HardwareProfile.h* file:

```
// Description: SD-SPI Chip Select Output bit
 #define SD_CS            LATDbits.LATD9
 // Description: SD-SPI Chip Select TRIS bit
 #define SD_CS_TRIS       TRISDbits.TRISD9

 // Description: SD-SPI Card Detect Input bit
 #define SD_CD            0
 // Description: SD-SPI Card Detect TRIS bit
 #define SD_CD_TRIS   TRISAbits.TRISA14
// Description: SD-SPI Write Protect Check
// Input bit
#define SD_WE          0
// Description: SD-SPI Write Protect Check
// TRIS bit
#define SD_WE_TRIS           TRISAbits.TRISA14
```

As it turns out, the *SD_CD* and *SD_WE* signals are really never major players in the grand scheme of things. For instance, in the case of the PIC32 driver firmware, by default, the *SD_CD* line's logic level is never checked. It is simply assumed to be logically low (card present) by the driver. On the other hand, the *SD_WE* signal's logic level is tickled. To gain access to the microSD card, the microSD driver wants the *SD_CD* and *SD_WE* I/O pins to report a logically low state. We can guarantee this by forcing the microSD driver's logical view of the lines to a low level. You can see that the "forcing" is accomplished by simply defining the *SD_CD* and *SD_WE* signals as zeros. To keep strange things from happening, the TRIS I/O pin assignments for both *SD_CD* and *SD_WE* pins call out unused PIC I/O pin RA14.

The *SD_CS* signal is user selectable. Note that our choice of chip select pins for the microSD resides on port D, pin 9. The microSD driver firmware is looking for a physical I/O pin that is attached to the logical *SD_CS*



■ Schematic 1. There are lots of other peripherals connected to the seemingly unassigned PIC32MX I/O pins. Right now, we are only interested in the microSD interface.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/ index.php?/magazine/article/june2015_DesignCycle**.

label. So, whatever we define in terms of I/O pin associations to the microSD driver's logical labels is what the firmware will manipulate. The same concept holds true for the *SD_CD* and *SD_WE* I/O pin physical and microSD driver logical definitions.

You can see in **Schematic 1** that the first SPI portal (SPI1) is used to service the microSD card. Microchip's SPI driver that is associated with the microSD driver needs to know which SPI portal is being used. The method to this madness is that the SPI driver needs to assure the I/O directions of the SPI portal pins being used are correctly set. Thus, we give the SPI driver what it wants yet again in the *HardwareProfile.h* file:

```
// Description: The TRIS bit for the SCK pin
 #define SPICLOCK    TRISDbits.TRISD10
// Description: The TRIS bit for the SDI pin
 #define SPIIN              TRISCbits.TRISC4
// Description: The TRIS bit for the SDO pin
 #define SPIOUT             TRISDbits.TRISD0
```

I've never gotten the microSD drivers to work with the default SPI portal speed of 20000000. So, before we leave the *HardwareProfile.h* file, we need to change the SPI portal speed:

```
// Define the SPI frequency
 #define SPI_FREQUENCY             (10000000)
```

I arbitrarily chose the working 10 MHz SPI frequency out of desperation and, as Spock said, "Random chance seems to have operated in our favor." Later, I came across the official documentation of this required change in Microchip document TB3112.

In addition to informing the microSD driver about SPI feeds and speeds, we also need to define the main system clock frequency in terms that the microSD and SPI drivers can understand:

```
#define GetSystemClock()     (80000000ul)
#define GetPeripheralClock() (GetSystemClock())
#define GetInstructionClock()(GetSystemClock())
```

The system clock speed is set in hardware by the configuration fuses. This set of PIC32MX configuration fuses is laid out in the main application file:

```
#pragma config FPLLMUL  = MUL_20
  // PLL Multiplier
#pragma config FPLLIDIV = DIV_2
  // PLL Input Divider
#pragma config FPLLODIV = DIV_1
  // PLL Output Divider
#pragma config FPBDIV   = DIV_1
  // Peripheral Clock divisor
#pragma config FWDTEN   = OFF
  // Watchdog Timer
#pragma config WDTPS    = PS1
  // Watchdog Timer Postscale
#pragma config FCKSM    = CSDCMD
  // Clock Switching & Fail Safe Clock Monitor
#pragma config OSCIOFNC = OFF
  // CLKO Enable
#pragma config POSCMOD  = HS
  // Primary Oscillator
#pragma config IESO     = OFF
  // Internal/External Switch-over
#pragma config FSOSCEN  = ON
  // Secondary Oscillator Enable
#pragma config FNOSC    = PRIPLL
  // Oscillator Selection
#pragma config CP       = OFF
  // Code Protect
#pragma config BWP      = OFF
  // Boot Flash Write Protect
#pragma config PWP      = OFF
  // Program Flash Write Protect
#pragma config ICESEL   = ICS_PGx2
  // ICE/ICD Comm Channel Select
```

An RTCC (Real Time Clock Calendar) crystal is part of our design. The RTCC is optional. However, it would be nice to timestamp the files we create. There is not much work involved in getting a timestamp thanks to support provided by the PIC32 peripheral library (plib). We've already taken the first step in bringing up the PIC32MX795F512L's RTCC by enabling the secondary oscillator in the configuration fuses. To enable file timestamping, a *#define* entry in the *FSconfig.h* file — which is part of the Microchip Memory Disk Drive File System (MDDFS) — is also required:

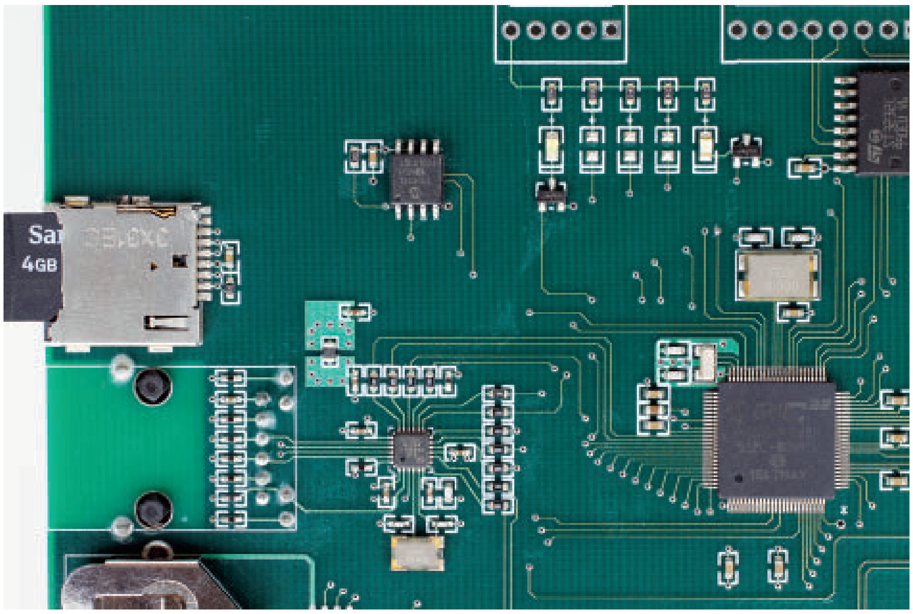
```
// Description: The USEREALTIMECLOCK macro will
// configure the code to generate timestamp
// information for files from the RTCC module.
// The user must enable and configure the RTCC
// module before creating or modifying files.
#define USEREALTIMECLOCK
```

Setting up the PIC32MX's RTCC is not a difficult task. However, it does require a lot of bit manipulation in a number of registers. Instead of pouring over RTCC operation in the PIC32MX datasheet, we will depend on the PIC32MX peripheral library RTCC macros to put the key into the RTCC's ignition switch and put it into gear:

```
// Initialize the RTCC
RtccInit();
// wait for the SOSC to be actually running and
// RTCC to have its clock source
while(RtccGetClkStat()!=RTCC_CLK_ON);
// set date and time
RtccOpen(0x10073000, 0x15041802, 0);
```

All of the *RtccXXXX* calls used to kick off the RTCC hardware are part of the PIC32MX peripheral library. With the assumption that we will be successful, the format of the *RtccOpen* function arguments will become clearer when you see the file timestamps that will be generated by our code.

With the introduction of Harmony, Microchip has decided to discontinue the PIC32MX peripheral library as an active and supported part of the 32-bit tool chain. Microchip will continue to provide the PIC32MX peripheral library as a download for support of legacy 32-bit designs.


■ Photo 1.The PIC32MX795F512L is loaded for bear as far as clocks and crystals are concerned. You can also see that the built-in Ethernet capabilities of the PIC are being fully utilized.

That does it for prepping the hardware via firmware.

To give you a physical perspective of the PIC and microSD hardware, I've snapped **Photo 1**.

Does It Work?

I'll bet it will. However, we need to twiddle some bits in the main project file. Normally, I like to have a serial port up and operational to assist in debugging. I have plans for that, but right now I only want to know if the microSD card code is working. So, instead of a full-blown serial port, we will settle for an LED indication at the successful completion of the program run. We will set that up like this:

```
TRISAbits.TRISA3 = 0;
LATAbits.LATA3 = 0;
```

The LED driven by I/O pin RA3 is identified as USERLED1 in **Schematic 1**. To help relieve the current load on the RA3 I/O pin, a NUD3105 relay driver buffers the current load used by USERLED1.

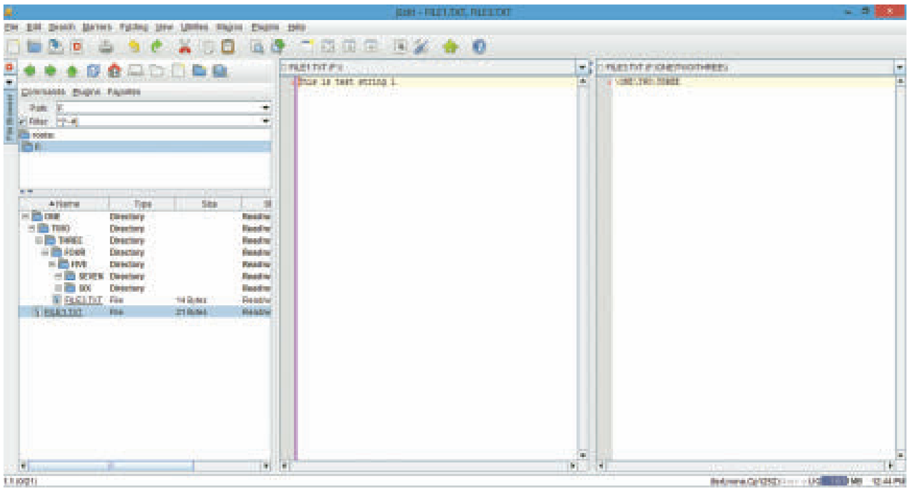
We will indicate a successful program by illuminating USERLED1. The code looks like this:

```
do{
        LATAbits.LATA3 = 1;
    }while(1);
```

There are three demos that you can choose from in the MLA project skeleton. I've decided to run *Demonstration1.c* with the modifications to I/O pin RA3 I've shown you. If all goes as planned, the demo program will build a directory tree that looks like this:

```
    \ -> FILE1.TXT
       -> ONE   -> TWO -> THREE   -> FILE3.TXT
                -> FOUR           -> FIVE -> SIX
                                         -> SEVEN
```

A more easily understood directory tree is shown in **Screenshot 1**.

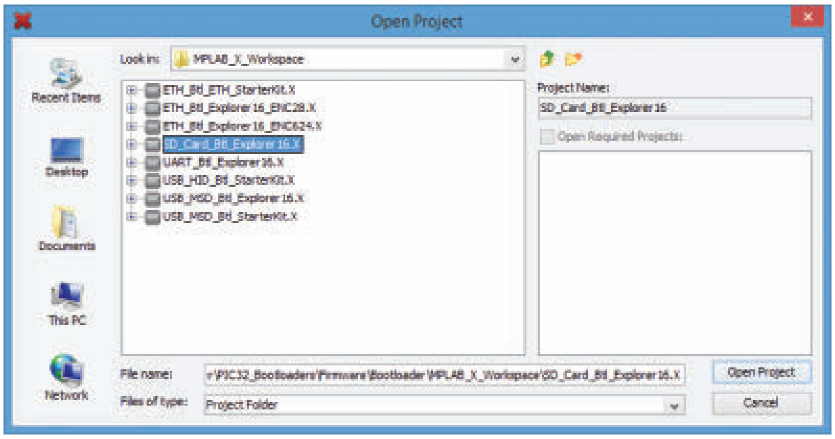

■ Screenshot 1. This is a screen capture of the microSD file tree. I used the services of the jEdit Programmer's Editor (free under the GNU General Public License) to also show the contents of the files that were created by the main application code.

By the way, as you can see in **Screenshot 1**, it works.

Easy PIC32MX Bootloading

So, microSD cards are not restricted to storing data. We can take everything we've discussed thus far and apply it to creating a microSD bootloader. To follow along, download the Microchip application note AN1388 and install the associated source code. Load up MPLAB X with the *SD_Card_Btl_Explorer16* project as shown in **Screenshot 2**.

Obviously, we aren't using an Explorer 16 dev board. It doesn't matter what the project is called, it's what we



Look in: MPLAB_X_Workspace

ETH_Btl_ETH_StarterKit.X
ETH_Btl_Explorer16_ENC28.X
ETH_Btl_Explorer16_ENC624.X
SD_Card_Btl_Explorer16.X
UART_Btl_Explorer16.X
USB_HID_Btl_StarterKit.X
USB_MSD_Btl_Explorer16.X
USB_MSD_Btl_StarterKit.X

Project Name: SD_Card_Btl_Explorer16

Open Required Projects:

File name: r\PIC32_Bootloaders\Firmware\Bootloader\MPLAB_X_Workspace\SD_Card_Btl_Explorer16.X

Files of type: Project Folder

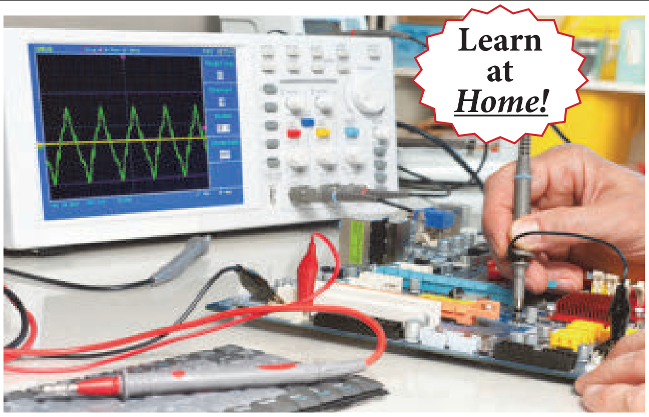
■ Screenshot 2. The Microchip project skeletons are easy to navigate as they are normally packaged within a meaningful directory structure.

do inside of the project that counts.

With that in mind, we will modify the necessary *HardwareProfile* include files (*HardwareProfile_PIC32MX_PIM_Explorer_16.h* and *HardwareProfile_PIC32MX_SD_PICtail.h*) to suit our PIC32MX hardware.

The first modification occurs in the Explorer 16 *include file*. In that modification code, we disable the PIC's JTAG interface and define the behaviors of USERLED1 and USERLED2. The pushbutton switch attached to I/O pin RA5 will be used to initiate a bootload request. Here's the conditional compilation segment that was modified:

```
#elif defined(__PIC32MX3XX_7XX__)
    #define mLED    LATDbits.LATD7
    #define BlinkLED() (mLED =
    ((ReadCoreTimer()
       0x0800000) == 0))
    #define InitLED() do{
\
     DDPCON = 0;                 \
     TRISDbits.TRISD7 = 0;       \
     LATDbits.LATD7 = 0;         \
    TRISAbits.TRISA3 = 0;        \
     LATAbits.LATA3= 0;          \
    }while(0)
// Switch ON all the LEDs to indicate Error.
#define Error()   LATAbits.LATA3 = 1;  \
                  LATDbits.LATD7 = 1;
#define ReadSwitchStatus() (PORTReadBits
```



Electronics Courses

Cleveland Institute of Electronics

Train at home to be a professional electronics or computer technician! Learn with hands-on labs, instructor support, online exams, videos and instructor led chat rooms.

FREE course catalog www.cie-wc.edu or (800) 243-6446

- NEW! Robotics Automation Lab
- Industrial Electronics with PLC
- Electronics with FCC
- Electronics Troubleshooting
- NEW! Computer Security Specialist
- Broadcast Engineering
- PC Troubleshooting
- Networking

Visit www.cie-wc.edu and start today!

CIE: 1776 E. 17th St., Cleveland, OH 44114 • (800) 243-6446 • Registration 70-11-0002H

The Easiest Way to Design Custom
Front Panels & Enclosures

You design it
to your specifications using our FREE CAD software, Front Panel Designer

We machine it
and ship to you a professionally finished product, no minimum quantity required

- Cost effective prototypes and production runs with no setup charges
- Powder-coated and anodized finishes in various colors
- Select from aluminum, acrylic or provide your own material
- Standard lead time in 5 days or express manufacturing in 3 or 1 days

FRONT PANEL EXPRESS

FrontPanelExpress.com

```
    (IOPORT_A, BIT_5) & BIT_5)
#endif
```

The changes made to the *SD_PICtail* include file are identical to the SPI I/O pin changes we made in the microSD driver *HardwareProfile.h*. Basically, the changes matched the SPI portal I/O pins to the actual hardware design.

The bootloader application actually checks the *SD_CD* signal. So, we're going to have to use that 10K pullup resistor you see attached to pin 10 of the microSD socket. That also means we can't define the *SD_CD* signal as a zero in the *SD_PICtail.h* include file.

If you take a look at the bootloader code, you will see that if we ignored the results of the *MDD_MediaDetect* function and a microSD card is not present in the microSD socket, the bootloader code would immediately go into an error condition.

By heeding the results of the *MDD_MediaDetect* function, an LED will blink when the microSD socket is empty and no application has been previously loaded. Here's a look at the code I just described:

```
//Initialize the media
while (!MDD_MediaDetect())
{
        // Waiting for media to
        // be inserted.
        BlinkLED();
    }

// Initialize the File System
    if(!FSInit())
    {
        // Indicate error and
        // stay in while loop.
    Error();
    while(1);
}
```

My LED is Blinking

I hope your microSD card LED blinks on cue, as well. AN1388 will walk you through all of the steps necessary to place a boot image on your microSD card and get it loaded onto your PIC32MX system. Be sure to check out TB3112, as well.

Adding microSD capability to any PIC32MX project is now an easy part of your Design Cycle. **NV**

Microchip Libraries for Applications
Harmony
AN1388
TB3112

PIC32MX795F512L
XC32 C Compiler
MBLAB X IDE

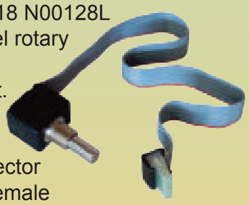
www.microchip.com

ALL ELECTRONICS
www.allelectronics.com
Order Toll Free 1-800-826-5432

3.5MM - 3.5MM AUDIO AUXILIARY CABLE
5' cable with 3.5mm stereo mini-phone plugs on each end. Flexible, good-quality cable, made for Motorola. Perfect for iPhone, Android & other audio connections.
CAT# CB-453
$1.40 each
50 for $1.25 each
100 for $1.00 each

1" SPEAKER IN ENCLOSURE
Great-sounding 8 Ohm speaker. Sealed 1.25" square x 0.90" deep enclosure. Mylar cone with rubber surround. 2.5" leads with 2-pin (2mm) socket connector.
CAT# SK-61
$2.00 each

ROTARY OPTICAL ENCODER
Bourns #ENA1J-518 N00128L
128PPR, 2-channel rotary optical encoder.
1/4" diameter shaft.
3/8-32 threaded bushing. 10" long radial ribbon connector with 5-conductor female connector, 0.10" spacing.
CAT# ENC-128
$28.00 each



MINI-SERVO W/ LINKAGE
Parkzone# PKZ1081 SV80. Miniature servo with horn and linkage.
Body: 24 x 12 x 15mm below mounting flanges. 400mm leads with 3-conductor female connector. Removed from equipment. Good condition.
CAT# DCS-111
$6.50 each

4-WIRE RESISTIVE TOUCH SCREEN FOR 8.4" LCD
SMK# NTX0100-8101R
CAT# TGL-100
$2.50 each

9VDC 0.66A POWER SUPPLY
GlobTek, Inc.
GTM41076-0609.
6' long cord with ferrite bead, and small custom coax plug, center+.
cULus.
CAT# PS-966
60 for $4.75 each
$5.25 each

63/37 SOLDER WIRE, 2 OZ
Leaded Solder Wire.
63/37 Tin/Lead no-clean.
0.020" (0.51mm) dia. 2 oz.
2.2% Rosin Flux Core.
CAT# SMD-5
$8.75 each

20K 2-ELEMENT SLIDE POT
Stanton# LF-304. Smooth, even slide action. 2.37" travel. Equipped with brushed aluminum face plate and rubberized, soft-touch knob. Prepped w/ a 5-pin (0.1") connector.
CAT# SLP-20KAX2B
10 for $6.00 each
$6.50 each

RARE EARTH MAGNET
Powerful neodymium magnet. Scalloped arc shape: 28mm x 16mm x 4.5mm.
CAT# MAG-148
70¢ each
10 for 65¢ each • 100 for 50¢ each

TECHFORUM
READER - TO - READER

All New TECHFORUM Online At www.nutsvolts.com/tech-forum

>>> QUESTIONS

Transistor As An STDT Switch

In a recent issue of *NV*, Roger Secura wrote the article "How to Use a Transistor as an SPST Switch." My question is how can an **STDT** switch be made using transistors, FETs, or other non-mechanical components? If it is possible, please post a circuit.

#6151 **Don E. Czyzyk**
Santa Clara, CA

SW Radio Info Needed

I am interested in building the shortwave radio shown in the schematic from Michael Williams Tech Forum question #2153 on page 78 of the February 2015 issue. However, I need more information, specifically the dimensions of L1 — length, diameter, etc., and the frequency range of the receiver.

I'm also seeking information on the coil data, size, number of turns, etc., for the shortwave and broadcast band coils for the Allied Space Spanner regenerative receiver, as well as modifications (plug-in coils VFO) that would increase the frequency coverage above and below the stock range of 6 to 12 MHz on the shortwave section of the receiver.

#6152 **Bradley Flener**
Central City, KY

Bipolar vs. MOSFET

I've read that bipolar transistors are current devices and MOSFET transistors — like old-fashioned vacuum tubes — are voltage-operated devices.

Although I understand the distinction conceptually, what does that mean from a practical perspective? For example, does this mean that bipolar are best for high power applications and MOSFETs are best for low voltage applications?

#6153 **Dale Schwan**
Douglas, ND

>>> ANSWERS

[#3153 - March 2015]
PCB Chem Disposal

I'm interested in photo etching copper-clad PCBs. Most guides don't say what to do with the chemicals when I'm done. Do I just pour them down the drain or will it hurt the environment (or my pipes)?

If you etch copper off a copper-clad PCB, you sure don't want to pour etchant into copper pipes! You don't want the copper ions from the etched board going into the waste-water system either. After all, we all live downstream of someone else. MG Chemicals offers two disposal ideas which you can read here: **www.mgchemicals.com/tech-support/ferric_faq/**.

Jon Titus
Herriman, UT

[#4152 - April 2015]
Test Lead Wire

Anyone know where I can purchase small quantities (25 ft rolls) of the different color jackets of good Beldon or (?) 65/30 test lead wire? I see it in 100 ft rolls $$, 10 colors, but that would be over a $1,000 for all 10.

Try Elenco WK-106 Hook-Up Wire kits which has six colors (red, yellow, black, white, green, blue) of 22 gauge wire in 25 foot rolls for around $15. See the website at **www.elenco.com**.

Tim Brown
Honea Path, SC

[#5151 - May 2015]
Replacement Plastic Transistors

Can someone recommend replacement plastic transistors (maybe a la PN2907) which are easily available, in order to substitute Q1, Q2, Q3, and Q4? I want to totally eliminate 2SB54 transistors WITHOUT any extreme changes to the original diagram for this (push-pull) phone amplifier I'm using.

#1 Mr. Franklin desires to reproduce a known circuit design that uses four 2SB54 transistors. The 2SB54 is an obsolete part. Furthermore, it is a germanium device. Mr. Franklin asks if a 2N2907 or similar silicon PNP device could be used instead.

Substituting silicon devices for germanium devices is a non-trivial exercise:

- The no-signal emitter-base junction voltage of germanium devices is about 0.3V, while that of silicon devices is about 0.7V. In a linear circuit such as that presented here, bias voltage appropriate to the transistor material type must be applied to overcome the junction voltage offset.
- Germanium transistors exhibit current leakage. This is often sufficient to provide self-biasing. Silicon devices have negligible leakage in the circuit under consideration, so external biasing must be considered.

The simplest thing to do is to replace the 2SB54 with another germanium part. The NTE100 device

Send all questions and answers by email to **forum@nutsvolts.com**
or via the online form at **www.nutsvolts.com/tech-forum**

— manufactured by NTE Electronics — has characteristics that are similar to those of the 2SB54. It is available from multiple distributors including Newark Electronics, Fry's Electronics, and Online Components, or you might try eBay. Pricing runs from about four to seven dollars per device. A datasheet — which includes a basing diagram — is available from NTE Electronics.

Peter A. Goodwin
Rockport, MA

#2 The swap you suggest will require some adjustment of transistor bias. The 2SB54 are germanium material with 0.15V to 0.2V base emitter bias. The PN2907 and similar are silicon with a 0.65V to 0.7V base emitter bias requirement. R2 and R6 will likely need to be increased in value. Quick estimate for R6 at 0.7V is around 1.2K. The collector feedback biasing makes the calculation of the R2 value more difficult and might not have to be changed due to the feedback.

Search for similar circuits using silicon transistors to get some better resistor values. Note that Q3 and Q4 require a little forward bias to prevent crossover distortion. I would expect considerable distortion if the biasing resistors are not modified to account for differences in germanium vs. silicon material GWS

George Shaiffer
Colorado Springs, CO

#3 The circuit is well designed with feedback in the first stage to stabilize its current and an emitter/resistor to limit the current in Q2. The output transistors, Q3 and Q4 are class B so draw no current until driven by audio. I simulated the Q1, Q2 circuit and it works perfectly with no changes using 2N2907 transistors. The Q2 collector current is 4.8 mA which is sufficient to drive the output transistors to watts of power.

Russell Kincaid
Milford, NH

#4 A 2N3906 should work okay. It has the same general characteristics as the 2SB54 transistors, except for being silicon instead of germanium, greater power dissipation, and slightly higher hfe (100 as opposed to 80).

Gene Sellier
Fairhope, AL

■LOOK FOR ■SEARCH FOR ■FIND
Find your favorite advertisers here!

ADvertiser INDEX

For Dads and Grads!

We Put The FUN In Electronics!

Super-Pro FM Stereo Radio Station

- ✔ **PLL synthesized for drift-free operation**
- ✔ **Built-in mixer - 2 line inputs and one microphone input, line level monitor output!**
- ✔ **Frequency range 88.0 to 108.0, 100 kHz steps**
- ✔ **Precision active low-pass "brick wall" audio filter!**
- ✔ **Dual LED bar graph audio level meters!**
- ✔ **Automatic adjustable microphone ducking!**
- ✔ **Easy to build through-hole design!**

This professional synthesized transmitter is adjustable directly from the front panel with a large LED digital readout of the operating frequency. Just enter the setup mode and set your frequency. Once selected and locked you are assured of a rock stable carrier with zero drift. The power output is continuously adjustable throughout the power range of the model selected. In addition, a new layer of anti-static protection for the final RF amplifier stage and audio inputs has been added to protect you from sudden static and power surges.

Audio quality is equally impressive. A precision active low-pass brick wall audio filter and peak level limiters give your signal maximum "punch" while preventing overmodulation. Two sets of rear panel stereo line level inputs are provided with front panel level control for both. Standard unbalanced "RCA" line inputs are used to make it simple to connect to the audio output of your computer, MP3 player, DVD player, cassette deck or any other consumer audio source. Get even more creative and use our K8094 below for digital storage and playback of short announcements and ID's! In addition to the line level inputs, there is a separate front panel microphone input. All three inputs have independent level controls eliminating the need for a separate audio mixer! Just pot-up the source control when ready, and cross fade to the 2nd line input or mic! It's that simple! In addition to the dual stereo line inputs, a stereo monitor output is provided. This is perfect to drive studio monitors or local in-house PA systems.

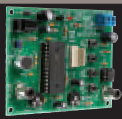
The FM100B series includes an attractive metal case, whip antenna and built in 110/220VAC power supply. A BNC connector is also provided for an external antenna. Check out our Tru-Match FM antenna kit, for the perfect mate to the FM100B transmitter.

We also offer a high power kit as well as an export-only assembled version that provides a variable RF output power up to 1 watt. The 1 watt unit must utilize an external antenna properly matched to the operating frequency to maintain a proper VSWR to protect the transmitter. *(Note: The FM100B and FM100BEX are do-it-yourself learning kits that you assemble.*

The end user is responsible for complying with all FCC rules & regulations within the US or any regulations of their respective governing body. The FM100BWT is for export use and can only be shipped to locations outside the continental US, valid APO/FPO addresses or valid customs brokers for documented end delivery outside the continental US).

FM100B	**Super-Pro FM Stereo Radio Station Kit, 5uW to 25mW Output**	**$269.95**
FM100BEX	**Super-Pro FM Stereo Radio Station Kit, 5uW to 1W Output**	**$349.95**

Audio Recorder & Player



Record and playback up to 8 minutes of messages from this little board! Built-in condenser mic plus line input, line & speaker outputs. Adjustable sample rate for recording quality. 4-switch operation that can be remote controlled! Runs on 9-12VDC at 500mA.

K8094	**Audio Recorder/Player Kit**	**$32.95**

Water Sensor Alarm

This little $7 kit really shined during Sandy! Simply mount the alarm where you want to detect water level problems. When the water level rises it touches the contacts and the alarm goes off! Sensor can even be remotely located. Runs on a 9V battery (not included).

MK108	**Water Sensor Alarm Kit**	**$6.95**

12VDC Regulated Switching Supply

Go green with our new 12VDC 1A regulated supply. Worldwide input 100-240VAC with a Level-V efficiency! It gets even better, includes DUAL ferrite cores for RF and EMI suppression. All this at a 10 buck old wallwart price!

AC121	**12VDC 1A Regulated Supply**	**$9.95**

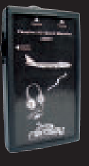
12VDC Worldwide Supply

It gets even better than our AC121 to the left! Now, take the regulated Level-V green supply, bump the current up to 1.25A, and include multiple blades for global country compatibility! Dual ferrite cores!

PS29	**12VDC 1.25A Global Supply**	**$19.95**

Passive Aircraft Monitor

PATENTED!



The hit of the decade! Our patented receiver hears the entire aircraft band without any tuning! Passive design has no LO, therefore can be used on board aircraft! Perfect for airshows, hears the active traffic as it happens! Available kit or factory assembled.

ABM1	**Passive Aircraft Receiver Kit**	**$89.95**

Signal Magnet Antenna

The impossible AM radio antenna that pulls in the stations and removes the noise, interference, and static crashes from your radio! Also helps that pesky HD AM Radio stay locked! Also available factory assembled.

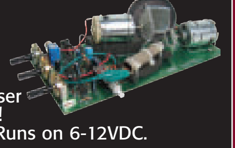
SM100	**Signal Magnet Antenna Kit**	**$89.95**

Laser Trip Sensor Alarm

True laser protects over 500 yards! At last within the reach of the hobbyist, this neat kit uses a standard laser pointer (included) to provide both audible and visual alert of a broken path. 5A relay makes it simple to interface! Breakaway board to separate sections.

LTS1	**Laser Trip Sensor Alarm Kit**	**$29.95**

Laser Light Show



Just like the big concerts, you can impress your friends with your own laser light show! Audio input modulates the laser display to your favorite music! Adjustable pattern & speed. Runs on 6-12VDC.

LLS1	**Laser Light Show Kit**	**$49.95**

The Learning Center!

Beginners To Advanced... *It's Fun!*

- ✔ **Learn, build, and enjoy!**
- ✔ **130, 200, 300, & 500 in one electronic labs!**
- ✔ **Practical through hole and SMT soldering labs!**
- ✔ **Integrated circuit AM/FM radio lab!**
- ✔ **AM, AM/FM, and SWL radio labs!**
- ✔ **Radio Controlled (RC) car!**
- ✔ **Beginner's non-soldering kits!**

For over 4 decades we've become famous for making electronics fun, while at the same time making it a great learning experience. As technology has changed over these years, we have continued that goal!

PL130A Gives you 130 different electronic projects together with a comprehensive learning manual describing the theory behind all the projects.

PL200 Includes 200 very creative fun projects and includes a neat interactive front panel with 2 controls, speaker, LED display and a meter.

PL300 Jump up to 300 separate projects that start walking you through the learning phase of digital electronics.

PL500 The ultimate electronics lab that includes 500 separate projects that cover it all, from the basics all the way to digital programming.

SP1A Whether young or old, there's always a need to hone your soldering skills. Either learn from scratch or consider it a refresher, and end up with a neat little project when you're done!

SM200K Move up to Surface Mount Technology (SMT) soldering, and learn exactly how to solder those tiny little components to a board!

AMFM108K We not only take you through AM and FM radio theory but we guide you through IC's. When you're done you've built yourself an IC based AM/FM radio that works great!

AM2 Learn the complete theory of AM broadcast radio and end up with a highly sensitive AM radio receiver!

AMFM7 Step up to AM/FM with this multi-band lab and learn the basics of both bands. The FM tuner is factory assembled and aligned!

SR3 Enter the world of SWL with this short-wave radio learning lab covering 6-8MHz and 12-18MHz!

AK870 One of the most exciting electronic learning kits that the kids will love! Build a complete RC speedster from the ground up! 7 remote functions!

PL130A	**130-In-One Lab Kit**	**$39.95**
PL200	**200-In-One Lab Kit**	**$84.95**
PL300	**300-In-One Lab Kit**	**$109.95**
PL500	**500-In-One Lab Kit**	**$249.95**
SP1A	**Through Hole Soldering Lab**	**$9.95**
SM200K	**SMT Practical Soldering Lab**	**$22.95**
AMFM108K	**AM/FM IC Lab Kit & Course**	**$36.95**
AM2	**AM Radio Learning Lab**	**$11.95**
AMFM7	**AM/FM Radio Learning Lab**	**$12.95**
SR3	**Short Wave Radio Learning Lab**	**$16.95**
AK870	**RC Speedster Car Kit**	**$29.95**

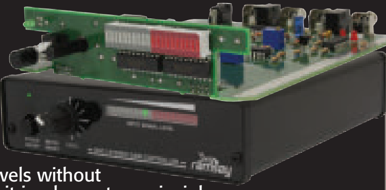
ramsey
www.ramseykits.com

There's only so much room on these two pages, so check it all out in our new virtual electronic catalog! Flip through the pages and search with ease! Visit ***www.ramseycatalog.com***

Follow Us and SAVE $$

Follow us on your favorite network site and look for a lot of super deals posted frequently... exclusively for our followers!

Stereo Audio Platform Gain Controller



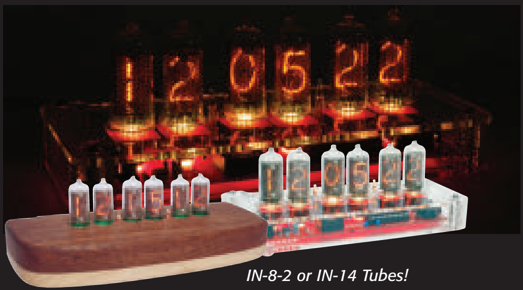
- ✔ Stereo audio processing while preserving audio dynamics!
- ✔ True stereo control keeps virtual sonic source location intact!
- ✔ Auto-bypass restores original levels when power is turned off!
- ✔ Built-in bar graph indication of signal level with display mute!

The SGC1 is one of our latest kits, and provides a great solution to the age-old problem: how can we easily correct inconsistent audio levels without negatively affecting the dynamics of the audio signal? The SGC1 circuit implements a principle known as the "Platform Gain Principle," which was originally developed by CBS Labs to allow transmitted audio levels to be automatically adjusted to keep them within a desired range.

Think of it like an audio engineer, constantly adjusting the output level in order to limit highs that would be too loud while boosting lower levels so that they can still be heard. You may think "oh, this is just another limiter/compressor!" Not so! Here's the real trick: keeping the full dynamic range ratio of the output signal the same as the original input - something the typical limiter/compressor can only dream of doing! The SGC1 can be placed in just about any standard analog stereo line level audio circuit to keep the audio level within the desired range. It's also the perfect addition to any of our hobby kit transmitters, allowing you to match levels between different audio sources while keeping lows audible and preventing the highs from overdriving.

In addition to its useful basic function and great audio performance, the SGC1 also boasts a front panel LED meter to give an indication of the relative level of the input signal, as well as a front level control that allows you to adjust the controller to the min/max center point of your desired level range. And yes, it is a ***Stereo Gain Controller!*** Meaning that the levels of both the left and right channels are monitored and adjusted equally, thereby maintaining the relative virtual position of things like instruments, singers and speakers! The entire unit is housed in a slim attractive black textured aluminum case that is sure to complement your studio or home theatre. If you're looking for perfect audio levels, hire a broadcast audio engineer, but if that doesn't fit your budget, the SGC1 is the next best thing! Includes 15VDC world-wide power adapter.

SGC1 Stereo Audio Platform Gain Controller Kit $179.95



IN-8-2 or IN-14 Tubes!

Classic Nixie Tube Clocks

Our next generation of classic Nixie tube clocks perfectly mesh today's technology with the Nixie era technology of the 60's. Of course, features you'd expect with a typical clock are all supported with the Nixie clock... and a whole lot more! The clocks are programmable for 12 or 24 hour mode, various AM/PM indications, programmable leading zero blanking, and include a programmable alarm with snooze as well as date display, 4 or 6 tube, kit or assembled!

We then jumped the technological time line of the 60's Nixie displays by adding the latest multi-colored LEDs to the base of the Nixie tubes to provide hundreds of illumination colors to highlight the glass tubes! The LED lighting can be programmed to any color and brightness combination of the colors red, green, or blue to suit your mood or environment. Then we leaped over the technological time line by integrating an optional GPS time base reference for the ultimate in clock accuracy! The small optional GPS receiver module is factory assembled and tested, and plugs directly into the back of the clock to give your Nixie clock accuracy you could only dream of! The clocks are available in our signature hand rubbed Teak & Maple or clear acrylic bases.

NIXIE Classic Nixie Tube Clock Kits From $229.95

RF Preamplifier

The famous RF preamp that's been written up in the radio & electronics magazines! This super broadband preamp covers 100 KHz to 1000 MHz! Unconditionally stable gain is greater than 16dB while noise is less than 4dB! 50-75 ohm input. Runs on 12-15 VDC.

SA7 RF Preamp Kit $16.95

Touch Switch

Touch on, touch off, or momentary touch hold, it's your choice with this little kit! Uses CMOS technology. Actually includes TWO totally separate touch circuits on the board! Drives any low voltage load up to 100mA. Runs on 6-12 VDC.

TS1 Touch Switch Kit $9.95

Mad Blaster Warble Alarm

If you need to simply get attention, the "Mad Blaster" is the answer, producing a LOUD ear shattering raucous racket! Super for car and home alarms as well. Drives any speaker. Runs on 9-12VDC.

MB1 Mad Blaster Warble Alarm Kit $9.95

Optically Isolated Module

The hobbyist's headache solver! Converts any AC or DC signal to logic level. The beauty is that the input and output are totally isolated from each other! Output can drive up to 150mA at 40VDC.

OM2 Optically Isolated Module Kit $16.95

Electronic Siren

What an attention getter! Just hook up a speaker, battery and a switch. Close the switch for an upward wail and release it for the downward return. One of our most popular minikits! Runs on 9V battery.

SM3 Electronic Siren Kit $7.95

Automatic VOX Voice Switch

This popular VOX switch provides a switched output when it "hears" sound! Use it to key a transmitter, turn on lights, or anything... all controled by your voice! Runs on 6-12VDC.

VS1 Automatic VOX Switch Kit $9.95

Logic Interface Module

Interface your digital output to the real world with an on-board SPDT relay rated at 240V at 10A! It takes a digital low (.5VDC or less) or a high (+1 to +12VDC) and provides your choice of an active low or high closure! It's that simple! Runs on 12VDC.

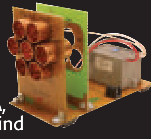
RI1 Logic Interface Kit $17.95

HV Plasma Generator

Generate 2" sparks to a handheld screwdriver! Light fluorescent tubes without wires! This plasma generator creates up to 25kV at 20kHz from a solid state circuit! Build plasma bulbs from regular bulbs and more! Runs on 16VAC or 5-24VDC.

PG13 HV Plasma Generator Kit $64.95

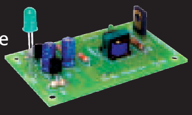
Air Blasting Ion Generator



Generates negative ions along with a hefty blast of fresh air, all without any noise! The steady state DC voltage generates 7.5kV DC negative at 400uA, and that's LOTS of ions! Includes 7 wind tubes for max air! Runs on 12-15VDC.

IG7 Ion Generator Kit $64.95

Tickle-Stick Shocker



The kit has a pulsing 80 volt tickle output and a mischievous blinking LED. And who can resist a blinking light and an unlabeled switch! Great fun for your desk, "Hey, I told you not to touch!" Runs on 3-6 VDC.

TS4 Tickle Stick Kit $9.95

Speedy Speed Radar Gun

Our famous *Speedy* radar gun teaches you doppler effect the fun way! Digital readout displays in MPH, KPH, or FPS. You supply two coffee cans! Runs on 12VDC or our AC121 supply.

SG7 Speed Radar Gun Kit $74.95

Tri-Field Meter Kit

"See" electrical, magnetic, and RF fields as a graphical LED display on the front panel! Use it to detect these fields in your house, find RF sources, you name it. Featured on CBS's Ghost Whisperer to detect the presence of ghosts! Req's 4 AAA batteries.

TFM3C Tri-Field Meter Kit $74.95

Stereo Ear Super Audio Amp

This "Stereo Ear" amp is one of the neatest and handiest high gain amps you will find! Dual high sensitivity electret mics are amplified 50x to provide the ultimate stereo source! Output is 3.5mm jack, runs on 3ea AAA's.

MK136 Stereo Ear Audio Amp Kit $9.95

Broadband RF Preamp

Need to "perk-up" your counter or other equipment to read weak signals? This preamp has low noise and yet provides 25dB gain from 1MHz to well over 1GHz. Output can reach 100mW! Runs on 12 volts AC or DC or the included 110VAC PS. Asmb.

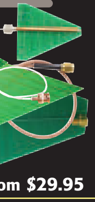
PR2 Broadband RF Preamp $69.95

Code Practice Oscillator

A great starter kit for young and old! Learn kit building while building a great little CW practice oscillator to learn the code! Built-in key, built in speaker, and audio tone adjust! Runs on a 9V battery.

CPO3 Code Practice Oscillator Kit $17.95

Precision PC Plane Antennas



Our LPY series PC antennas continue to be the favorite for virtually all RF and wireless applications. From microwave links, wireless mics, to RFID, we've got you covered. Check our site for details!

LPYSeries Precision PC Plane Antennas from $29.95

Active Receive Antenna

The EASY antenna, regardless of frequency! This active RX antenna performs like a 60' long wire. Works on all bands, and provides over 15dB of gain. Features auto-bypass when turned off. Runs on 9V battery.

AA7C Active Receive Antenna Kit $59.95

GET THE NUTS AND VOLTS DISCOUNT!

*Mention or enter the coupon code **NVRMZ142** and receive 10% off your order!*

800-446-2295
www.ramseykits.com

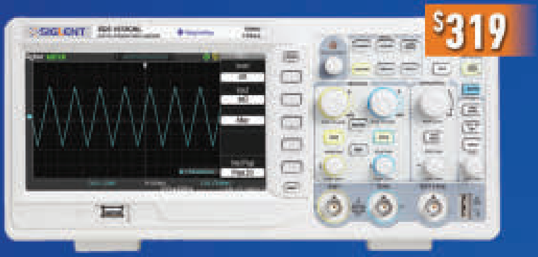
Prices, availability, and specifications are subject to change. Not responsible for typos, stupids, printer's bleed, or missing graduation caps! Robin says give your dad a neat electronic kit instead of another soap on a rope in a wooden gift box! Visit www.ramseykits.com for the latest pricing, specials, terms and conditions. Copyright 2015 Ramsey Electronics®.. so there!

RAMSEY ELECTRONICS®
590 Fishers Station Drive
Victor, NY 14564
(800) 446-2295
(585) 924-4560

SIGLENT

5% EDUCATIONAL DISCOUNT!

THE BEST VALUE IN TEST EQUIPMENT



SDS1072CML OSCILLOSCOPE

70MHz, 2 ch, unique EasyHunting™ waveform capture. 1GSa/s 2Mpt memory, built-in 6 digit counter.

SDG805 AWG GENERATOR

5MHz, 1 ch, 125MSa/s EasyPulse™ technology for low jitter, fast rising/falling edges at every frequency.

SDS2104 OSCILLOSCOPE

100MHz, 4 ch, 2GSa/s, 28Mpt memory, 110,000 wfm/s capture, huge 8" color LCD, AWG/Logic analyzer options.

SHS806 HANDHELD OSCILLOSCOPE

60MHz, 1GSa/s, 2Mpt memory portable scope with features from expensive bench-top instruments.

SPD3303D POWER SUPPLY

High precision 195 watt supply, 3 outputs: 30V@3A x2, 2.5-5V@3A, 4 digit voltage / 3 digit current display.

SDM3055 MULTIMETER

$5^{1/2}$ digit, 150 readings/sec with math and statistical functions and remote communication interfaces.

INCREDIBLE LOW PRICES, EXCELLENT CUSTOMER SERVICE, FREE TECH SUPPORT

(585) 385-1750 WWW.SAELIG.COM INFO@SAELIG.COM

Saelig.com unique electronics